RL-TR-96-174
Final Technical Report
November 1996

MULTIPROCESSOR IMPLEMENTATION OF A REAL-TIME CELP ALGORITHM

University of Kansas

Glenn Prescott, Hari N. Chakravarthula,
Sinivas Sivaprakasam, and Timouthy Johnson

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-96-174 has been reviewed and is approved for publication.

APPROVED:

STEPHEN C. TYLER
Project Engineer

FOR THE COMMANDER:

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/C3BB, 525 Brook Road, Rome, NY 13441-4505. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 1996	Final Apr 94 - Oct 95

4. TITLE AND SUBTITLE
MULTIPROCESSOR IMPLEMENTATION OF A REAL-TIME CELP ALGORITHM

5. FUNDING NUMBERS
C - F30602-94-C-0104
PE - 62702F
PR - 4519
TA - 42
WU - PG

6. AUTHOR(S)
Glenn Prescott, Hari N. Chakravarthula,
Sinivas Sivaprakasam, and Timouthy Johnson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Telecommunications and Information Sciences Laboratory
CECASE
University of Kansas
Lawrence, KS 66045

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory/C3BB
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-96-174

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Stephen C. Tyler/C3BB/(315) 330-3618

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
The objective of this effort was to develop the methodology of transferring communication signal processing algorithms from a single processor (SUN Workstation) to multiple DSP processors TMS320C40s. The CELP (Code-Excited Linear Prediction) voice compression algorithm was chosen for real-time implementation of 3 processors. The algorithm operates at an output rate of 4800 bits per second with an input sampling rate of 8000 samples per second. Using efficient parallel processing algorithms (optimized to reduce the inter-processor communication time overhead) one can implement complicated communication functions such as the CELP algorithm on more than one processor and achieve real-time performance. A block diagram description of the CELP algorithm has been developed using the Signal Processing Worksystem (SPW), a block oriented design tool from the Alta Group of Cadence to generate optimized code for a VMEbus based network of TMS320C40 processors. The CELP algorithm is based on the U.S. Federal Standard 1016. Data transfers between the processors is achieved by using the C40 processors' high speed communication ports and concurrent multi-channel DMA transfer capability. The approach frees the CPU of burdensome interprocessor communication functions.

14. SUBJECT TERMS
CELP, Digital Signal Processing, SPW

15. NUMBER OF PAGES
96

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

The purpose of this report is to present the implementation of a real-time CELP (Code-Excited Linear Prediction) speech compression algorithm on a network of three TMS320C40 DSP processors. The algorithm operates at an output rate of 4800 bits per second with an input sampling rate of 8000 samples per second. Using efficient parallel processing algorithms (optimized to reduce the inter-processor communication time overhead) one can implement complicated communication functions such as the CELP algorithm on more than one processor and achieve real-time performance.

A block diagram description of the CELP algorithm has been developed using the Signal Processing Worksystem (SPW), a block oriented design tool from the Alta Group of Cadence to generate optimized code for a VMEbus based network of TMS320C40 processors. The CELP algorithm is based on the U.S. Federal Standard 1016. Data transfers between the processors is achieved by using the C40 processors' high speed communication ports and concurrent multi-channel DMA transfer capability. This approach frees the CPU of burdensome inter-processor communication functions. In addition to decreasing CPU execution time, the use of multiple processors reduces the memory requirements, and thus a larger code-book can be used.

Our implementation takes about 30 cycles (nearly 20 assembly instructions) for a single frame transfer (240 samples) between two processors. The entire CELP encoder takes an estimated 500,000 cycles per frame on the C40 while the decoder is much less computationally intensive. In this report, we present the various algorithmic and code-level optimizations used for the real-time implementation of the CELP algorithm and the resulting performance characteristics. Related issues such as parallel processing are also discussed.

Contents

Chapter 1

Introduction

Coding at low bit rates while still maintaining high quality is of considerable interest in current speech research with potential applications to satellite communication, cellular-radio communication and secure communication for both military and commercial applications. A simple digital representation of speech is Pulse Code Modulation (PCM) [1], which is obtained by bandlimiting and sampling the speech signal at its Nyquist frequency and representing the amplitudes of the samples by a finite number of values (quantization). For telephone applications, the analog speech signal is bandlimited to 4 KHz and sampled at 8 KHz. The resulting sample amplitudes are adequately represented with 13 bits/sample. By logarithmically compressing the input samples, this number is reduced to 8 bits/sample without compromising speech quality. This compression is called μ-law or A-law PCM [1] depending on the compression scheme used.

In many applications, such as mobile communications and voice storage applications, the available channel capacity is limited. Even in applications where there is no hard limit on the channel capacity, it is economical to find signal representations that have the least number of bits. The bit rate can be reduced by representing the sample amplitudes with fewer bits. A coarse quantization increases the quantization error, since speech samples can take a wide range of

possible values (a large dynamic range). Speech samples, however, exhibit a strong correlation from sample to sample, and for voiced speech signals, such as vowels, there is a strong correlation between adjacent periods. Removal of these correlations reduces the dynamic range of the sample amplitudes , allowing more efficient quantization. Linear Predictive Coding (LPC) [2] is an efficient procedure for removing these correlations. The linear prediction method predicts the current input sample value based on previously reconstructed values. The difference between the current value and its predicted value is quantized and transmitted. The procedure of generating the difference signal and using this signal, or its quantized version, to produce the reconstructed signal can be described as linear filtering with filters based on the predictors. The input is filtered through an analysis filter, and the difference signal forms the excitation for the synthesis filter. Since the speech signal characteristics vary over time, the prediction is made more efficient by adjusting the predictors periodically. This procedure is known as adaptive prediction and is used in 32 kbps adaptive differential PCM (ADPCM) [1].

At lower bit rates only a few bits per sample are available for encoding the excitation signal, and the key issue in designing coders for these rates is finding bit-efficient representations. Instead of trying to find a representation that matches the difference signal, it is more efficient to find an excitation sequence that for the given synthesis filter produces a signal that is close to the original input signal. Each sample of the excitation signal affects many samples of reconstructed speech because of the recursive structure of the synthesis filters. Therefore the choice of the excitation signal is made by measuring its effect on the reconstructed speech over more than one sample. In other words, the decision about the best quantized representation is not made instantaneously but is delayed for an interval that includes several samples. This approach is called *delayed decision coding*. Since this decision depends on the error between the original and reconstructed speech, it requires synthesis during analysis, and the procedure is generally referred to as

analysis-by-synthesis predictive coding.

Code-excited linear prediction (CELP) coders use another approach to reduce the number of bits per sample. Here, both the encoder and decoder store a collection of C possible sequences of length L in a codebook. The excitation for each frame is described completely by the index to an appropriate vector in the codebook. This index is found by an exhaustive search over all possible codebook vectors and the selection of the one that produces the smallest error between the original and reconstructed signals. The computational complexity of the CELP algorithm is very high and a lot of methods have been thought about to reduce the number of computations involved and perform the coding real-time. This neccessiates optimization at both the algorithmic and code levels. The CELP algorithm that was developed was implemented real-time on a network of TMS320C40 processors. The features of the C40 such as dedicated parallel communications ports(commports) and DMA channels allowed us to develop an efficient protocol for data transfers between processors. A test system was implemented on three C40 processors. Chapter 2 provides a background on the production of speech and defines some basic terms associated with speech coding. Chapter 3 provides an overview of the CELP algorithm and summarizes some fast algorithms. Chapter 4 deals with the system level implementation issues on a network of TMS320C40 processors. Chapter 5 addresses the issues of parallel processing on multiple processors. The results obtained are presented in Chapter 6. The conclusions drawn from this work and pointers to future work in this area are presented in Chapter 7.

Chapter 2

Background

2.1 Basic Definitions

In voiced speech, the vocal chords vibrate in a quasi-periodic way. Speech segments with voiceless excitation are generated by turbulent air flow at a constriction or by the release of a closure in the vocal tract. The parameters we have to determine are the manner of excitation- *i.e., the presence of a voiced excitation and the presence of a voiceless excitation, and, for the segments of the speech signal where a voiced excitation is present, the rate of vocal cord vibration, which is usually referred to as pitch determination or fundamental frequency determination.*

The pitch, i.e., fundamental frequency, has a key position in the acoustic speech signal. The prosodic information of an utterance is predominantly determined by this parameter. The ear is an order of magnitude more sensitive to changes of fundamental frequency than to changes of other speech signal parameters. The quality of the vocoder is essentially influenced by the quality and faultlessness of the pitch measurement. For a number of reasons, the pitch determination is one of the more difficult tasks in speech analysis. Some of the reasons are outlined below:

- Speech is a non-stationary process; the momentary position of the vocal

tract may change abruptly at any time. This leads to drastic variations in the temporal structure of the signal, even between subsequent pitch periods.

- For an arbitrary speech signal uttered by an unknown speaker, the fundamental frequency can vary over a range of almost four octaves(50 to 800 Hz)

- The excitation signal is not always regular. Even under normal conditions, the glottal waveform exhibits occasional irregularities. In addition, the voice may temporarily fall into vocal fry or creak, which is a nonpathologic mode of voice excitation with rather large and irregular intervals between subsequent glottal pulses.

- Additional problems arise in speech communication systems, where the signal is often distorted or band limited (for instance, in the telephone channel).

2.1.1 Pitch: Basic Definitions

Pitch can be measured in many ways. Since the speech signal is nonstationary and time variant, aspects of strategy such as the starting point of measurement, the length of the measuring interval, the way of averaging, or the operating domain (time, frequency, lag etc) of an individual algorithm start to influence the results and may lead to estimates that differ from algorithm to algorithm.

There are three points of view for looking at a speech processing problem: the *production*, the *signal processing*, and the *perception* points of view. In the actual case of pitch determination the production point of view is obviously oriented toward the generation of the excitation signal in the larynx; we will thus have to start from a time domain representation of the waveform as a train of laryngeal pulses. If an algorithm or device works in a speech-production oriented way, it measures individual *laryngeal excitation cycles* or, if some averaging is performed, it determines the *rate of vocal fold vibration.* The signal processing point of view

can be characterized in such a way that (quasi-) periodicity is observed in the signal, wherever that signal comes from, and the task is just to extract the features that best represent this periodicity, The pertinent terms are *fundamental frequency* and *fundamental period.* If individual cycles are determined, we may speak of *pitch periods.* The perception point of view leads to frequency domain representation, since pitch sensation corresponds to a frequency and not to an average period or a sequence of periods. This point of view is associated with the original meaning of the term *pitch.*

Defining the different representations of pitch, it appears reasonable to proceed from production to perception. Going in that direction we will start at a local and detailed representation and arrive at a more global representation in the case of the perception-oriented view. The basic definitions could thus be as follows:

T_0 (the pitch period) is defined as the elapsed time between two successive laryngeal pulses. Measurement starts at a well-specified point within the glottal cycle, preferable at the point of glottal closure or - if the glottis does not close completely - at the point where the glottal area reaches its minimum.

The corresponding frequency domain definition reads as follows:

F_0 is defined as the fundamental frequency of an (approximately) harmonic pattern in the (short-term) spectral representation of the signal. It depends on the particular method whether F_0 is calculated as the frequency of certain harmonic divided by the respective harmonic number m (including m=1), as the frequency difference between adjacent spectral peaks, or as the greatest common divisor of the frequencies of the individual harmonics.

The perception point of view of the problem leads to a different definition of

pitch. Pitch perception happens in the frequency domain.

F_0 is defined as the frequency of the sinusoid that evokes the same perceived pitch (residue pitch, virtual pitch etc) as the complex sound which represents the input speech signal.

This definition is principally different from the previous ones. Above all, it is a *long-term* definition. The pitch perception theories were developed for stationary complex sounds and were extended toward short pulse trains with varying amplitude patterns and constant frequencies but not toward signals with varying fundamental frequencies.

The algorithms used for determining the pitch are referred to as *pitch determination algorithms*, (PDAs).

2.1.2 Categorizing the various principles of pitch determination

The existing PDA principles can be split up into two gross categories: *time domain* PDAs and *short-term analysis* PDAs.

In any short-term analysis PDA a *short-term (or short-time) transformation* is performed in a preprocessor step. The speech signal is split up into a series of frames as shown in Figure 2.1; an individual frame is obtained by taking a limited number of consecutive samples of the signal x(n) from the starting point, n=q-K+1, to the ending point, n=q. The frame length, K, is chosen short enough so that the parameter(s) to be measured can be assumed approximately constant within the frame. On the other hand, K must be large enough to guarantee that the parameter remains measurable. For most short-term analysis PDAs a frame thus requires two or three complete periods at least. In extreme cases, when F_0 changes abruptly or when the signal is irregular, the contradiction of these two

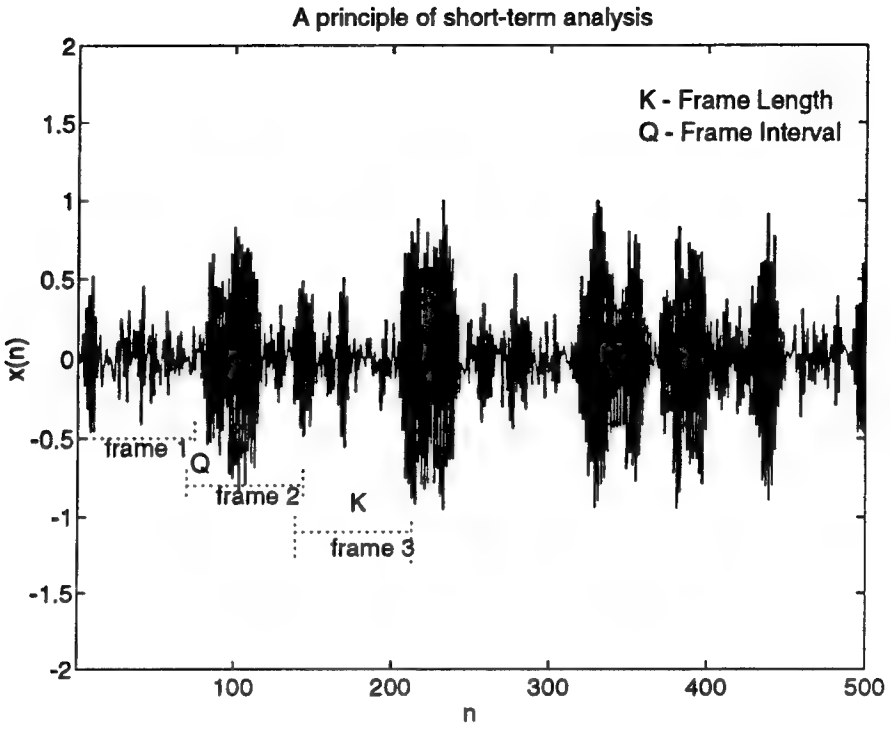

Figure 2-1: A principle of short-term analysis

conditions can be a source of error. The frame interval Q, *i.e.*, the distance between consecutive frames (or its reciprocal, the frame rate), is determined in such a way that any significant parameter change is documented in the measurements.

The short-term transformation, so to speak, is intended to behave like a concave mirror which focusses all the scattered information on pitch, as far as it is available within the frame, into one single peak in the spectral domain. This peak is then determined by a peak detector (as the usual implementation of the basic extractor in this type of PDA). Hence the output signal of the basic extractor is a sequence of average pitch estimates. The short-term transform causes the phase relations between the spectral domain and the original signal to be lost. At the same time, however, the algorithm loses much of its sensitivity to phase distortions and signal degradation. Unfortunately, the increased reliability of the

algorithm is accompanied by increased computing effort (which is at least one order of magnitude higher than for a time domain PDA). Much of this effort goes into the numeric calculation of the transform. Besides the search for reliability, the search for fast implementation has therefore been an important issue in the design of short-term analysis PDAs.

Not all the spectral transforms show the desired focussing effect. Those ones which do are in some way related to the power spectrum: correlation techniques, frequency domain techniques, and a least-squares approach. Among the correlation techniques we find the well-known autocorrelation function, which became successful in pitch determination of band-limited signals when it was combined with time domain center clipping.

If the signal were strictly periodic, the distance function would take on a value of zero at d=T_0. For the quasi-periodic speech signal there will be a strong minimum in the PDF at this value of the lag(delay time) d. In contrast to all the short-term PDAs, where the estimate of T_0 or F_0 is indicated by a maximum whose position and value have to be determined, the minimum has an ideal target value of 0 so that we need only determine its position. For this reason, distance functions do not require (quasi-) stationarity within the measuring interval; they can cope with very short frames of one pitch period or even less. This principle thus represents the only short-term analysis PDA which is able to follow definition.

The frequency domain methods are also split up into two groups. Direct determination of F_0 as the location of the lowest peak in the power spectrum is unreliable and inaccurate. It is thus preferred to investigate the harmonic structure of the signal. One way to do this is spectral compression, which computes the fundamental frequency as the greatest common divider of all harmonics. The power spectrum is compressed along the frequency axis by a factor of two, three, etc. and then added to the original power spectrum. This operation gives a peak at F_0 resulting from the coherent additive contribution of the higher harmonics.

Some of these PDAs stem from theories and functional models of pitch perception in the human ear.

The second frequency domain approach leads back to the time domain. Instead of transforming the power spectrum itself (which would lead to the autocorrelation function) however, the inverse transform is performed on the logarithmic power spectrum. This results in the *cepstrum*, which shows a distinct peak at the "quefrency" (lag) d=T_0.

Finally there is the least-squares (maximum likelihood) approach. This was originally a mathematical procedure for separating a periodic signal of unknown period T_0 from Gaussian noise within a finite signal. Since the speech signal is not periodic and the background noise (plus the aperiodic components of the speech itself) cannot be expected to be Gaussian, the approach has to be modified slightly in order to work in a PDA.

In summary, short-term analysis PDAs provide a sequence of average pitch estimates rather than a measurement of individual periods. There are not very sensitive to phase distortions and to absence of the fundamental harmonic. On the other hand, computing effort is very high.

Chapter 3

The CELP Algorithm

3.1 Overview of the CELP algorithms

The CELP algorithm falls into a general class of coders to which the multi-pulse linear predictive coding (MPLPC) and regular-pulse linear predictive coding (RPLPC) algorithms also belong. This class of coders operate on sampled speech on a frame by frame basis. When the original code excited linear predictive coding (CELP) algorithm was developed, it was found to provide good speech quality at intermediate bit rates (4.8-9.6 kb/s). However this speech quality was obtained at the expense of very high computational complexity, making real-time implementation on low-cost hardware impossible. To remedy this problem, a variety of fast procedures are available. This project has made use of a number of fast algorithms for facilitating real-time implementation and the following sections deal with specific algorithms.

The CELP algorithm operates on sampled speech on a frame by frame basis. A filter is used to describe the spectral envelope of the speech signal. The coefficients of the filter are obtained using the linear prediction (LP) technique. They are quantized so that the same filter can be constructed at both the transmit and receive ends of the channel. The excitation for the filter is determined using an

analysis-by-synthesis procedure. A set of candidate excitation sequences is stored in a codebook, and synthetic speech is generated using each of these sequences. The index of the sequence producing the most accurate speech is then transmitted to the receive end of the channel.

For the encoding of the excitation signal, it is useful to first compute a target excitation sequence t for the current frame. The target excitation sequence is defined as the vector which will drive the synthesis LP filter to produce the current-frame speech signal. At the outset of the determination of a target excitation sequence, the delay line of the (all-pole) synthesis filter is assumed to contain the last synthetic speech samples of the previous frame. Thus, the excitation vector t differs from the residual speech signal produced by a whitening filter using the same LP coefficients because it compensates for quantization errors which occurred in previous frames.

The filtering operation of the excitation sequence by the all-pole filter can be performed by the convolution of the excitation sequence t with the impulse response of the filter. For a single frame, this convolution can be written as a matrix multiplication Ht, where H is an N x N lower triangular Toeplitz matrix containing the impulse response h_i of the filter in its first column.

$$H = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \cdots & \cdot \\ \cdot & h_1 & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & h_0 \; 0 \\ h_{N-1} & h_{N-2} & \cdots & h_1 \; h_0 \end{bmatrix} \tag{3.1}$$

As will be shown later, it is often advantageous to truncate the impulse response after a sufficient number (R) of samples. If z is the zero-input response of

the synthesis filter for the current frame, the speech signal s can be written as:

$$s = Ht + z \tag{3.2}$$

While the MPLPC algorithm quantizes the target vector t by the sequential determination of a set of pulse locations and amplitudes, the CELP algorithm encodes the target vector via a shape-gain vector quantization. The codebook used in this vector quantization can consist of stochastic or deterministic sequences, or sequences which were obtained from a training procedure.

The goal is to accurately model the speech signal s, and not the target excitation t. Matching the target excitation directly has been used, but in general, this will result in a significant degradation in performance. Thus, an exhaustive search through synthetic speech vectors generated by each of a set of optimally scaled candidate vectors $\mu^{(i)}x^{(i)}$ is performed. Such an analysis-by-synthesis search procedure is computationally much more expensive than a search involving a simple comparison of excitation signals.

The matrix H can be interpreted as a weighting of the $x^{(i)}$ and t vectors in the computation of the least squares error criterion which evaluates the accuracy of the synthetic speech signal. In fact, the selection of the best candidate vector is a shape-gain vector quantization of the target excitation vector using the following dynamic error criterion:

$$\epsilon^{(i)} = (t - \mu^{(i)}x^{(i)})^T H^T H(t - \mu^{(i)}x^{(i)}) \tag{3.3}$$

where $\mu^{(i)}$ is the optimal scaling factor for the candidate vector $x^{(i)}$. $\mu^{(i)}$ is defined as:

$$\mu^{(i)} = \frac{t^T H^T H x^{(i)}}{x^{(i)T} H^T H x^{(i)}} \tag{3.4}$$

This expression for the optimal scaling factor $\mu^{(i)}$ can be substituted into 3.3, but the resulting elegant error criterion is only of practical value if the quantization of $\mu^{(i)}$ is performed after the search procedure. It is more accurate to use the quantized value of $\mu^{(i)}$ during the search process.

The spectral weighting introduced by the matrix $H^T H$ into the vector quantization of the target vector t results in an equal expectation value for the error over the entire spectrum of the synthetic speech signal. It is known, however, that audible quantization noise is masked by the formants of the speech signal, suggesting that more of this quantization noise should be put under the formants. To diminish the perceived noise signal resulting from the vector quantization procedure, the relative weighting of the formant regions in the error criterion can be decreased by deemphasizing the spectrum, here represented by the H matrix. This can be accomplished by moving the poles of the LP filter inward by a constant factor, usually denoted by γ, for which a value of around 0.8 is appropriate. The weighting is easily implemented by multiplying the filter coefficients of the LP filter by powers of this factor. Hereafter, we assume that this perceptual weighting is included in the search procedure.

The H matrix of 3.1 results in an error criterion with a symmetric weighting matrix $H^T H$. Additional symmetry of the error criterion is beneficial in the development of fast algorithms. The spectral weighting matrix $H^T H$ becomes a Toeplitz matrix if the impulse response h_i, truncated after R samples, is used in its entirety for each sample of the excitation vector. In most situations, it is appropriate to use a value of R less than N, since the perceptual weighting reduces the effective length of the impulse response. In such cases, $H^T H$ becomes a band matrix (in addition to being symmetric and Toeplitz). The matrix H is now:

$$H = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ h_{R-1} & h_{R-2} & \cdots & \cdot \\ 0 & h_{R-1} & \cdots & \cdot \\ \cdot & 0 & \cdots & h_0 \\ \cdot & 0 & \cdots & h_1 \\ \cdot & \cdot & \cdots & \cdot \\ 0 & \cdot & \cdots & h_{R-1} \quad h_{R-2} \\ 0 & \cdot & \cdots & 0 \quad h_{R-1} \end{bmatrix} \tag{3.5}$$

Equation 3.3 can still be employed to evaluate the error criterion with this modified H matrix. The matrix H is still N x N, but the vectors $H(t - \mu^{(i)}x^{(i)})$ are now of length $N + R - 1$.

The error criterion employing 3.1 is called the covariance approach, while the modified criterion, which employs 3.5 is called the autocorrelation approach (since the symmetric Toeplitz H^TH contains the autocorrelation of the truncated impulse response in its first column). The autocorrelation error criterion results in a more equitable weighting of the samples of the excitation vector. The major advantage of the modification of the error criterion is the symmetry created in the H^TH matrix, which is now a symmetric Toeplitz matrix; and if $R < N$, a symmetric Toeplitz band matrix. In contrast, if 3.1 is used in the error criterion, then the H^TH matrix is symmetric (band) matrix.

CELP algorithms are usually applied with a pitch loop, which takes advantage of the periodicity of the (voiced) speech signal to provide more efficient encoding. Two basic procedures exist: the open and closed pitch loops. In the open pitch loop, the underlying periodicity of the speech signal is determined from the autocorrelation function of the residual speech signal. The same long-term corre-

lation is then introduced to the synthetic excitation function using a one- or three-tap pitch loop. In the closed pitch loop, a cross-correlation between the target excitation and the synthetic excitation of previous frames is used to select the appropriate delay. The closed pitch loop can be interpreted as a search through a set of overlapping vectors, which correspond to the recent history of the synthetic excitation. The search finds that sequence in the recent history of the synthetic excitation which best matches the target excitation vector for the current speech frame. Therefore, the closed pitch loop can be seen as shape-gain vector quantization procedure using an adaptive codebook. In this light, the CELP algorithm becomes a two-stage shape-gain vector quantization of the target excitation, where the first stage uses an adaptive codebook and the second stage uses a fixed codebook (usually of stochastic nature). The analysis-by-synthesis aspect of the closed pitch loop results in better speech quality. Since the closed loop parameters (gain and delay) must be determined from a procedure similar to that of the second vector quantization stage, its computational cost per candidate vector is comparable. The main difference between the two procedures is that the codebook for the closed pitch loop is dynamic while the codebook of the second stage is fixed. The following discussion of fast procedures addresses both the dynamic codebook of the closed pitch loop and the fixed codebook of the second stage.

It is natural to use a two-stage vector quantization of the target excitation sequence. In general, when a multiple stage shape-gain vector quantization of this sequence is used, the error criterion described by 3.3 is applied at each stage. After the selection of the optimally scaled winning candidate vector $x^{(j)}$ for a particular quantization stage, the target for the next vector quantization stage is obtained by subtracting $\mu^{(j)}x^{(j)}$ from the current target vector. The resulting vector t - $\mu^{(j)}x^{(j)}$ is quantized in the next stage.

The quantization of the target vector t provides a special challenge because the error criterion contains the spectral weighting matrix $H^T H$. This weighting

makes the search through the codebook computationally more expensive, and the dynamic character of the $H^T H$ matrix prevents one from using some of the established techniques, such as tree searches, for minimizing the computational effort in vector quantization. This dynamic character means that the closest neighbors of a particular candidate vector in the codebook may be different from frame to frame, eliminating all fast methods that make use of the similarity of candidate vectors.

3.2 Baseline Computational Effort of the CELP Algorithm

In this section, a baseline is provided for the computational effort to which the various fast procedures can be compared. the computational method used to obtain this baseline effort is more general than the fast procedures. It consists of a straightforward evaluation of 3.3, and can be applied to codebooks which are stochastic, deterministic or trained, and dynamic or fixed in time. The computational effort required by the various versions of the CELP algorithm will be evaluated in terms of "operations". An operation is defined as a multiply, an add or a subtract, or, whenever possible, a multiply-accumulate. The latter operation is restricted to successive additions to a single scalar quantity and does not include the sequential addition of vectors. This count of computational effort is consistent with the architecture of modern floating-point digital signal processors devices, which can perform the defined operations in a single instruction cycle.

The evaluation of the cost of 3.3 is facilitated by first multiplying out all terms. It is seen that the first term of the resulting expression, $t^T H^T H t$, is a constant which is of no consequence during the search, and does not have to be computed. Equations 3.3 and 3.4 are both dependent on the cross-correlation term $t^T H^T H x^{(i)}$ and the energy term $x^{(i)T} H^T H x^{(i)}$. The evaluation of the error

criterion for each candidate vector consists of three parts: the computation of the cross-correlation term, the computation of the energy term, and the quantization of $\mu^{(i)}$. The division operation in 3.4 can be eliminated by cross multiplying for each comparison of the quantization process of $\mu^{(i)}$. If $\mu^{(i)}$ is quantized after the search procedure, a similar cross multiplication will have to be performed in the evaluation of 3.3. In both cases, the evaluation of the actual value of the scaling factor, and in particular the division, can be performed after the best candidate has been selected from the codebook and is of no significance in figuring the computational effort.

The quantization process of the scaling factor $\mu^{(i)}$ is most efficiently performed using a binary tree structure. The quantization will then require as many comparison operations as the number of bits assigned to the quantization of the scaling factor.

Computation of the cross-correlation $t^T H^T H x^{(i)}$ requires relatively little effort since the vector $H^T Ht$ can be computed first. This leaves only N operations per candidate vector for the remaining inner product, apart from overhead. For the covariance criterion, this fast evaluation of the cross-correlation term using the vector $H^T Ht$ can be interpreted as follows. The vector $y = Ht$ represents the speech signal less the zero-input response of the synthesis filter. The multiplication of the vector t by the lower triangular Toeplitz matrix H as defined in 3.3 is equivalent to performing a convolution. Similarly, the multiplication of y by an upper triangular Toeplitz matrix H^T with as its first row an impulse response h_i can be interpreted as the convolution of the temporally reversed version of the vector $H^T y$. This interpretation of the evaluation of the matrix multiplication $H^T y$ resulted in the designation "backward filtering" for this operation.

The energy term must be recomputed for each vector. Performing the full matrix multiplication in its most straightforward form would require $N(N+1)$ operations per candidate vector, but realizing that the H matrices are lower tri-

angular leads to considerable savings. In this case $N(N+3)/2$ operations are required to perform this inner product by first computing $\boldsymbol{Hx}^{(i)}$ and then evaluating the inner product. When added to the effort for the cross-correlation term, this leads to a total of $N(N+5)/2$ operations per frame. Using scalar notation, the common method to compute the energy and correlation terms is first to calculate the convolution $\boldsymbol{Hx}^{(i)}$ ($N(N+1)/2$ operations), then to find the inner product of this vector with itself to get the energy term(N operations), and finally to find the inner product of the same vector with $\boldsymbol{Ht}$ (N operations) to obtain the cross-correlation term. This will again require a total of $N(N+5)/2$ operations, but the actual operations performed are somewhat different from those used in the description of the matrix notation method. This computation effort will serve as the basis for our comparisons.

For an adaptive codebook of 256 entries, and a fixed stochastic codebook of 1024 entries, a frame length of 60 samples, and a sampling rate of 8000 Hz, $N(N+5)/2$ operations per frame translates into 333 million operations per second. In addition to these calculations, overhead computations will have to be performed, including the computation of the term $\boldsymbol{H}^T\boldsymbol{Ht}$, and, if optimal performance is required, quantization of the scaling factor $\mu^{(i)}$ inside the search loop. To put this effort into perspective, the fastest general purpose digital signal processing devices can perform about 40 million operations per second (TMS320C40). Thus, a significant reduction in computational effort is needed to make real-time implementations possible.

3.3 Fast Algorithms

This section will discuss the properties of methods which minimize the computational effort for the CELP algorithms. The algorithms have been classified into several groups: fast procedures which rely on specially designed codebooks, var-

ious multistage search procedures, procedures operating in transform space, the autocorrelation method, methods which use overlapping codebook entries, and procedures which perform the energy term calculations off-line.

3.3.1 Special Codebook designs

A very simple but effective method to reduce the computational effort can be obtained by center clipping the candidate vectors of a stochastic codebook. Center clipping results in significant savings in the computational effort if the vector $\boldsymbol{H}\boldsymbol{x}^{(i)}$ is computed by adding the responses due to each of the samples of $\boldsymbol{x}^{(i)}$ rather than by computing the output vector $\boldsymbol{y}^{(i)} = \boldsymbol{H}\boldsymbol{x}^{(i)}$ one sample at a time as in conventional filtering operations. Using the former procedure, and skipping the entire column of $\boldsymbol{H}$ whenever a sample of $\boldsymbol{x}^{(i)}$ is zero will result in a savings proportional to the number of zeros introduced for the $\boldsymbol{H}\boldsymbol{x}^{(i)}$. However, this column skipping means that the multiply-accumulate operation must be replaced by separate multiply and add operations. [3] has used codebooks which have 90-95 % of their entries set to zero. According to [4], a codebook where all samples are generated from independently identically distributed Gaussian processes performs marginally better when it is center clipped. For 90% center clipping, the computational effort will be on average $2.1N + 0.1N^2$ operations per codebook entry. The center clipped codebook can be further simplified by constraining the nonzero entries to be either -1 or 1. This provides improved speech quality compared to an i.i.d. Gaussian codebook. Usage of a sparse codebook provides a simple and effective way to reduce the computational effort required for the second quantization stage in CELP algorithms. Contrary to the results for the stochastic codebook, center clipping of the adaptive codebook reduces the synthetic speech quality and, therefore, this procedure is not recommended as a method for reducing the computational effort.

At higher bit rates, deterministic codebook design has been used to obtain fast

algorithms. Here, the excitation sequence is more directly linked to the transmitted index. For example, in the 13.5 kb/s version of the algorithm, the excitation sequence consists of the bits of the index, with the 0 replaced by -1. This version provides a one bit scalar encoding for each sample, resulting in low error sensitivity. The high level of correlation of the codebook entries is exploited in the fast algorithm.

3.3.2 Multistage searching

The spectral weighting of the error criterion of the CELP algorithm contributes most of the computational effort. Relaxing the spectral weighting can lead to significantly faster computations, at the expense of performance. Several authors have proposed to use relaxed spectral weighting in a preselection stage for the CELP algorithm.

Preselection methods for the candidate vectors which bypass the dynamic character of the criterion for the selection of the excitation vector were developed in [5], [6] and [3]. The first two preselect the candidate excitation vectors by performing a direct (least squares) comparison with the target excitation vector. Thus they use for preselection the same (non-optimal) error criterion which was used for the actual search. The full search procedure, including the spectral weighting, is then performed on a reduced set of candidate vectors. Although the method was performed on the stochastic codebook, it can be used on the adaptive codebook as well, and given the periodic structure of both the speech and the excitation signal the preselection process may actually have more merit here. However, no work is reported on such a procedure. The speech quality of this preselection method for the stochastic codebook improves by increasing the number of survivors of the preselection scheme. No information is provided suggesting the number of survivors required to make the method equivalent in quality to a full search.

In another preselection method, the codebook vectors are filtered off-line through

a small set of P filters representing spectral classes (thus limiting this method to the second quantization stage, where the codebook vectors do not change). During the operation of the coder, the spectrum of the current frame of speech signal is first classified as belonging to one of the spectral classes. The corresponding vectors which have been filtered off-line are then compared to the current speech. The K candidate vectors which perform best under this pre-selection criterion are then searched using the conventional error criterion. A disadvantage of this method is that it increases the storage requirements significantly. The speech domain vectors cannot be overlapped as the stochastic codebook entries can be, and for a codebook of M candidates, $M(P+1)N$ words of storage are required. This number can be lowered slightly if the candidate vectors are overlapped. The energy terms for the preselection stages can be stored. Then, the total number of operations for this method is $MN + KN(N+5)/2$ for each codebook search. For a stochastic codebook with 1024 entries a preselection to a subset of only 30 candidates using 4 spectral classes maintains the original quality.

Center clipping of the codebook does not reduce the computational effort of the preselection in the two algorithms discussed above. Checking for zero samples requires similar effort to that of the computation in this case. The effort required for the final spectral-weighted search over the selected set is reduced by center-clipping but this will not result in a dramatic decrease in overall computational effort.

The conventional two-stage vector quantization procedure (adaptive and stochastic codebook) can be extended to include multiple stochastic codebook quantization stages. At constant frame size this will reduce the computational effort dramatically. In general, replacing single stage searches with multiple stage searches results in less efficient encoding of a signal. Thus, speech quality should be carefully monitored when implementing this type of procedure. An advantage of the multistage procedures is that they do not put restrictions on the codebook design,

allowing training of the codebooks.

3.3.3 Transform Methods

The above methods modify the numerical results of the CELP algorithm. Modification of the codebooks may lead to improved speech quality, but nonoptimal searches throughout the codebook at best maintain the existing speech quality. There are several procedures which reduce the complexity of the algorithm without affecting its numerical results. Let us first look into the Singular Value Decomposition procedure. For this purpose we can write the error criterion as follows:

$$\epsilon = (y - \mu^{(i)} H x^{(i)})^T (y - \mu^{(i)} H x^{(i)}) \tag{3.6}$$

A singular value decomposition can now be performed on the H matrix such that $Hx^{(i)} = UDVx^{(i)}$, where U and V are unitary matrices and D is diagonal. Thus 3.6 can be rewritten as:

$$\epsilon = (U^T y - \mu^{(i)} DV x^{(i)})^T (U^T y - \mu^{(i)} DV x^{(i)}) \tag{3.7}$$

where the unitary property of U was used. If the statistical distribution of the samples of $x^{(i)}$ is independent, identical, and Gaussian, then so is that of $Vx^{(i)}$ (since V is unitary). Given this we can perform a search using the following error criterion:

$$\epsilon = (U^T y - \mu^{(i)} D u^{(i)})^T (U^T y - \mu^{(i)} D u^{(i)}) \tag{3.8}$$

where u are the vectors of the stochastic codebook, which represent the transform-

domain excitation vectors. The expression for the optimal scaling factor $\mu^{(i)}$ is :

$$\mu^{(i)} = \frac{y^T U D u^{(i)}}{u^{(i)T} D^2 u^{(i)}} \tag{3.9}$$

3.8 and 3.9 can be combined if the factor $\mu^{(i)}$ is quantized after the search procedure, but it is advantageous to quantize 3.9 before the evaluation of the error criterion 3.8. When the best candidate vector $\mu^{(j)}$ in the transform domain has been found, the actual candidate vector can be obtained from $x^{(j)} = V^T u^{(j)}$. As before the bulk of the computation is contained in the evaluation of the energy term $u^{(i)T} D^2 u$ and the cross-correlation term $y^T U D u$. The minimal number of operations required is $3N$ (compute Du and complete the inner product with itself, and then evaluate its inner product with $U^T y$).

By modifying the notation, the singular value decomposition method can be described as an eigenvalue decomposition method. Let us write the error criterion as a weighted vector quantization of the target excitation vector t as :

$$\epsilon = (t - \mu^{(i)} x^{(i)})^T H^T H (t - \mu^{(i)} x^{(i)}) \tag{3.10}$$

The eigen value decomposition of the matrix $H^T H = V^T \Gamma V$ results in:

$$\epsilon = (Vt - \mu^{(i)} u^{(i)})^T \Gamma (Vt - \mu^{(i)} u^{(i)}) \tag{3.11}$$

where the optimal scaling factor $\mu^{(i)}$ is:

$$\mu^{(i)} = \frac{t^T V^T \Gamma u^{(i)}}{u^{(i)T} \Gamma u^{(i)}} \tag{3.12}$$

Computation of the energy and cross-correlation terms again requires $3N$ operations per candidate vector.

The eigenvalue decomposition and SVD methods have as a major disadvantage the large amount of overhead required to perform the eigenvalue decomposition or SVD. This can be reduced by using the autocorrelation error criterion and applying the methods provided in ref f, which take advantage of the symmetry of the modified $H^T H$ matrix.

The eigenvalue decomposition and SVD methods rely on the fact that an arbitrary unitary transform does not change the character of the codebook. Thus, the method can be used for i.i.d. gaussian sequences but not for deterministic, center-clipped or trained codebooks.

Further savings can be obtained by performing the search only in those dimension s of the transform domain that contribute to the time domain function. For vectors of length 40, half of the components of the transform domain can be eliminated from the search with negligible effect on objective performance. This is the case because most speech has strong short-term correlation, resulting in a large range of eigenvalue magnitudes. This in turn means that many components can be neglected. In fact, that error can be reduced by zeroing those components which are not optimized. However, the audible performance of the CELP algorithm deteriorates more quickly when zeroing points of the excitation vectors in the eigen transform domain than from zeroing samples of excitation vectors which are not transformed. This suggests that the approximation of the excitation function (after removal of the long-term correlation by the closed pitch loop) by a sequence of i.i.d. Gaussian samples is not accurate. Some significant structure is present in the excitation sequence after removal of the pitch, which is more

difficult to model in the eigen transform domain, but is at least partly described by center clipping in the time domain.

Another transform method takes advantage of the Fourier transform. It is based on the fact that convolution in the time domain becomes multiplication in the frequency domain. Since this relation does not hold for truncated sequences, the autocorrelation error criterion must be used to take advantage of the Fourier transform. The vector length for the transform operation must be $N + R - 1$ samples. The entries of any fixed codebook can be transformed into the Fourier domain, including those obtained from a training procedure.

If we denote the discrete Fourier transform of $x_p^{(i)}$ by $X_k^{(i)}$, h_i by H_k, and t_p by T_k then we obtain for the new error criterion:

$$\epsilon = \sum_{k=0}^{k=N+R-1} (T_k - \mu^{(i)} X_k^{(i)})^* H_k^* H_k (T_k - \mu^{(i)} X_k^{(i)}) \tag{3.13}$$

where * indicates transpose conjugate, and where

$$\mu^{(i)} = \frac{\sum_{k=0}^{k=N+R-1} Re(X_k^{(i)*} T_k H_k^* H_k)}{\sum_{k=0}^{k=N+R-1} X_k^{(i)*} X_k^{(i)} H_k^* H_k)} \tag{3.14}$$

Again, the main terms to be computed are the cross-correlation term and the energy term. Computation of $X_k^{(i)*} X_k^{(i)}$, $H_k^* H_k$, and $\mathrm{Re}(X_k^{(i)} T_k)$ requires 2 operations each. taking into account the symmetry of the sequences, we obtain a total of $3(N + R - 1)$ operations per candidate. It is likely that the computational effort for the Fourier transform, like that of the SVD method, can be further reduced by considering only those dimensions which contribute to the time domain output.

The transform procedures provide and optimal search through a codebook. Neither the eigenvalue (and SVD) nor the Fourier transform procedures can be

used to compute the closed pitch loop, since the candidate sequences must be available in the transform domain. More important is the problem of computational overhead which the methods require. This is high for the eigenvalue decomposition and SVD methods, but lower for the DFT method where a fast Fourier transform can be employed if $N + R - 1$ is chosen conveniently, In addition, the Fourier transform procedure is capable of dealing with any type of fixed codebook (including trained codebooks) while the eigenvalue and SVD methods work for specific codebooks only. However, it is noted that the core computations of the energy and cross-correlation terms require more effort in the case of the DFT ($3(N+R-1)$ per candidate). In a typical case, this will be a 50% increase in the computational effort.

3.3.4 The Autocorrelation Method

The autocorrelation method is an efficient method to compute the energy term. It makes use of the fact that the sum of the squares of the convolution of two sequences equals the cross correlation of the autocorrelation of these two sequences. For this relationship to hold, the convolution cannot be truncated, requiring, approximately, the usage of the autocorrelation error criterion. If the energy term is written in scalar notation, and if advantage is taken of the symmetry of the autocorrelation functions, it is seen that

$$R_0^{(h)} R_0^{(x)} + 2 \sum_{k=1}^{k=R-1} R_k^{(h)} R_k^{(x)} \sum_{k=0}^{k=N+R-1} \left(\sum_{m=0}^{m=R-1} h_m x_{k-m} \right)^2 = R_0^{(h)} R_0^{(x)} + 2 \sum_{k=1}^{k=R-1} R_k^{(h)} R_k^{(x)} \qquad (3.15)$$

where it is implicitly assumed that $x_m = 0$ for $m < 0$, and where $R_k^{(h)}$ is the kth element autocorrelation of h.

If the autocorrelation sequences for all the excitation candidates are stored, the evaluation of the energy term is very fast. The autocorrelation procedure requires only R operations per candidate vector to compute the energy term; adding to the N operations for the cross-correlation term results in a total of $N + R + 1$ operations per candidate. This is a very small computational effort, but unlike other procedures it cannot be lowered further by center clipping. The only overhead is the computation of the autocorrelation function of the impulse response H_i.

The autocorrelation procedure does not put any constraints on the type of fixed codebook used, and can be applied to trained codebooks. The method implements an optimal search through the particular codebook. A disadvantage of the autocorrelation method is its requirement for storage of the first R values of the autocorrelation sequences for all of the codebook entries. This makes this method useful for a search through a fixed codebook only and leads to an increase in the storage requirements for such a codebook. The increased storage is particularly significant when compared to overlapping stochastic codebooks. The storage of the autocorrelation function requires MR storage locations, regardless of the particular form of the codebook.

3.3.5 Method using overlapping codebook entries

The use of overlapping codebooks provides several advantages. Obviously the storage requirements are drastically reduced for such codebooks. In addition, the dependency of the neighboring candidate vectors can be exploited in the derivation of fast algorithms. Further, since the adaptive codebook is inherently of the same nature (assuming the pitch is greater than the frame length), any of these procedures can be applied to the adaptive codebook as well. The method can be extended for the case where the pitch is smaller than the frame length.

On the negative side, an overlapping codebook cannot accommodate a trained

set of candidate vectors. and although the discussed fast procedures implement an optimal search throughout the overlapping stochastic codebook, the dependency introduced by the overlapping character can negatively influence the performance of algorithms. With increasing codebook size, fewer shifts are required to maintain the performance level of a fully independent codebook. The level of overlap (and thus computational effort) versus performance will require evaluation at the design stage of the coder. Usually a shift of two samples between adjacent candidate vectors will be appropriate.

Since the neighboring candidate vectors of overlapping codebooks are related to each other by shifts and endpoint corrections, it is useful to define the shifting matrices:

$$S_f = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & 0 & 1 \\ 0 & . & . & . & 0 & 0 \end{bmatrix} \tag{3.16}$$

$$S_b = \begin{bmatrix} 0 & 0 & 0 & . & . & 0 \\ 1 & 0 & 1 & . & . & . \\ . & 1 & . & . & . & . \\ . & . & . & . & 0 & 0 \\ 0 & . & . & . & 1 & 0 \end{bmatrix} \tag{3.17}$$

and the masking matrix I_k

$$I_k = \begin{bmatrix} 1 & 0 & . & . & . & 0 \\ 0 & 1 & 1 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & 1 & 0 & . \\ 0 & . & . & 0 & 1 & . \\ . & . & . & . & . & . \\ 0 & . & . & . & . & 0 \end{bmatrix} \tag{3.18}$$

where the last nonzero element appears in the kth row and column.

A comparison of expressions for the responses for $Hx^{(i)}$ and $Hx^{(i+1)}$ suggests a quick method to perform the computation of the energy term for an overlapping codebook. If the neighboring candidates overlap for all but one samples and we start with $Hx^{(i)}$, we can obtain $Hx^{(i+1)}$ by adding the response due to the last sample of $x^{(i+1)}$ and subtracting the response due to the first sample of $x^{(i)}$:

$$Hx^{(i+1)} = S_f Hx^{(i)} + H(I - I_{N-1})x^{(i+1)} - HI_1 x(i) \tag{3.19}$$

where we assume the extended H matrix of 3.5 is used, so that the full responses are preserved. A shift by one sample requires $2R$ multiply and $2R$ add operations. The autocorrelation error criterion can be computed from the resulting vectors of length $N + R - 1$. Thus, this method requires $2(N + R - 1) + 4LR$ operations for the evaluation of the autocorrelation error criterion, where L is the number of shifts between adjacent candidate vectors.

For the covariance error criterion the effort required by the previous method can be reduced by moving backwards through the candidate vectors. In this case the response due to the new sample needs to be added, but the subtraction can be removed (it is taken care of by the shift).

$$Hx^{(i+1)} = S_b Hx^{(i+1)} + H(I_1)x^{(i)} \tag{3.20}$$

If we add the computations for the inner product, a total of $N+2LR$ operations is required per candidate vector for the computation of the energy term of the conventional error criterion. Including the cross-correlation term we obtain a total of $2N + 2LR$.

The endpoint correction procedure can be further accelerated by center clipping the codebook (applicable only to fixed codebooks). The evaluation of the second term on the right hand side of 3.20 can be eliminated whenever the new excitation sample $I_1 x^{(i)}$ vanishes. In this case the energy term is identical to that of the previous candidate vector with the contribution of the last sample of $Hx^{(i)}$ subtracted. The subtraction requires 1 operation. It is also seen that if both endpoint points are zero, no correction has to be applied, but including this test will actually increase the computational requirements of the algorithm. For 90 % sparsity the overall effort is then $N + 0.1(N + 2LR) + 0.9$ per candidate.

The above procedures are recursions for the vector $Hx^{(i)}$. A disadvantage is that it cannot take advantage of the multiply-accumulate operation of modern DSP chips. as as result, the speed of the procedure is further enhanced if the ternary codebook is used. The computational effort is then $2N + LR$ operations per candidate.

3.4 Description of the CELP algorithm used

The CELP algorithm used was developed for 4800 bps operation [7]. A basic block diagram of a CELP coder is shown in Figure 3.1. The algorithm is based on the work in [8] and [9] and the US Federal Standard 1016. The CELP analysis consists of three basic functions: 1) short-term linear prediction, 2) long-term

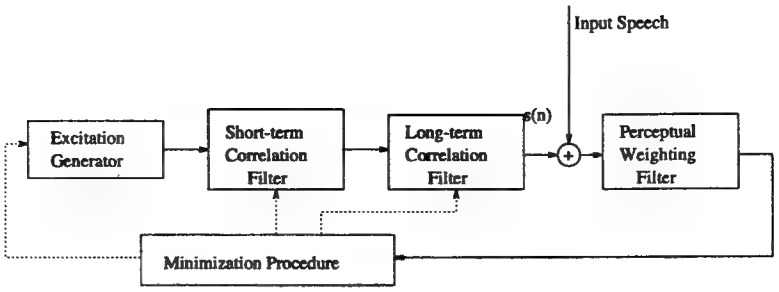

Figure 3-1: Analysis by synthesis adaptive predictive coder

adaptive code book search, and 3) innovation stochastic code book search. CELP synthesis consists of the corresponding three synthesis functions performed in reverse order with the optional addition of a fourth function, called a postfilter, to enhance the output speech. The transmitted parameters are the stochastic code book index and gain, the adaptive code book index and gain, and 10 line spectral parameters (LSP).

The speech signal is segmented into a sequence of frames for the evaluation of the filter that models the envelope of the speech short-time spectrum. The duration of the speech frame is 30 msec in this work.

For each subframe of the speech frame, the excitation needs to be determined. Typically, each prediction frame is broken into four excitation subframes. Accordingly, the subframe is 7.5 msec.

The reconstructed speech is produced by filtering the output signal from the excitation generator through both a long-term predictor synthesis filter and a short-term predictor synthesis filter. The excitation signal is found by minimizing the weighted mean-squared error over several samples, where the error signal is obtained by filtering the difference between the original and reconstructed signals through a weighting filter. Both short-term and long-term predictors are adapted over time, the short-term predictor being adapted at a slower rate than the long-term predictor. The analysis operation in the encoder involves local synthesis

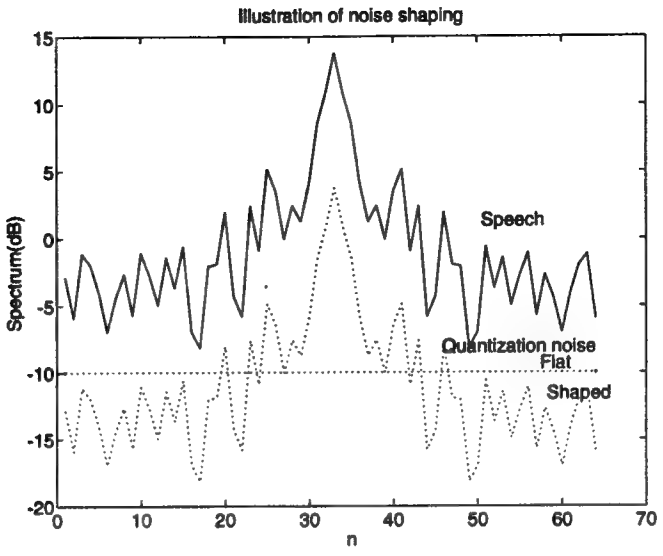

Figure 3-2: Illustration of the use of noise shaping to reduce loudness of coding noise

of the speech, the description of the analysis procedure completely describes the decoder.

Minimizing the mean squared error results in a quantization noise that has the same energy at all the frequencies of the input signal. The hearing system has only a limited capability to detect noise in the frequency bands in which the speech signal has high energy. To make use of this effect, the quantization noise has to be distributed in relation to the speech power over the different frequency bands. The desired distribution is shown in Figure 3.2. The noise weighting filter deemphasizes the energy of the error lying near the peak of the formants. The weighting filter may be defined as

$$W(z) = \frac{H_{lpc}(z/\gamma)}{H_{lpc}(z)} \tag{3.21}$$

where H_{lpc} is the short-time correlation filter transfer function and the bandwidth expansion factor γ is typically 0.8.

3.4.1 Synthesis

The CELP synthesizer is used in the receiver and transmitter to generate speech by a parallel gain-shape excitation of a linear prediction filter. The excitation is formed using a fixed stochastic code book and an adaptive code book in parallel. The stochastic code book contains sparse, overlapping, ternary-valued, pseudo-randomly generated codewords. Both code books are overlapped and can be represented as linear arrays, where each 60 sample codeword is extracted as a contiguous block of samples. In the stochastic code book, the codewords overlap by a shift of -2 (each codeword contains all but two samples of the previous codeword and two samples). The adaptive code book has a shift of one sample or less between its codewords. The codewords with shifts of less than one sample are interpolated and correspond to noninteger pitch delays. The linear prediction filter's excitation is formed by adding a stochastic code book vector, given by index i_{st} and scaled by a gain factor g_{st}. The adaptive code book is then updated by this excitation for use in the following subframe. Thus, the adaptive code book contains a history of past excitation signals. The delay indexes the codeword containing the best block of excitation from the past for use in the present. The number of samples back in time in which this vector is located is called the pitch delay or adaptive code book index. For delays less than the vector length, a full vector does not exist and the short vector is replicated to the full vector length to form a codeword. Finally, an adaptive postfilter is added to enhance the synthetic output speech.

3.4.2 Analysis

The transmitter's CELP analyzer, contains a replica of the receiver's synthesizer (except the postfilter) that, in the absence of channel errors, produces speech identical to that in the receiver. This generated speech, s^- is subtracted from the original speech signal. The difference signal is perceptually weighted. The search

procedure finds the adaptive and stochastic code book indices that minimize the perceptually weighted error.

3.4.3 Short-term Predictor

The short-term predictor filter models the short-term correlations (spectral envelope) of the speech signal and has the form

$$\frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.22}$$

where a_k are the short-term predictor coefficients and p is the order of the filter. The value of p used was 10.

The short-term predictor parameters are determined from the input speech (forward estimation) via the autocorrelation technique [17]. In this work, 34 bits are used to encode the 10 predictor coefficients using a 30 ms Hamming window and 15 Hz bandwidth expansion. The bandwidth expansion replaces the LPC coefficients a_is with $a_i\gamma^i$. This shifts the poles towards the origin of the Z-plane by the weighting factor γ. This bandwidth expansion of 15 Hz improves speech quality, is beneficial to LSP quantization [9] and also to fast direct conversion of predictor coefficients to quantized LSPs [10].

The predictor coefficients are updated every 30 msec. This sudden change can introduce transients in the reconstructed speech. To reduce these transitions it is useful to interpolate the short-term predictor parameters. This is done by converting the LPCs to LSPs (Line Spectral Pairs). The LSPs for the present frame are linear combinations of the LSPs computed for the previous frame and the newly computed LSPs for the present frame. The LSPs are coded using 34 bits.

3.4.4 Long-term Predictor/Adaptive Code book search

The long-term predictor filter models the long-term correlations (spectral fine structure) in the speech signal. It models primarily the periodicity in the excitation signal. It has the form

$$\frac{1}{P(z)} = \frac{1}{1 - \sum_{k=-(q-1)/2}^{(q-1)/2} b_k z^{-M+k}}, q = 1, 3, ... \tag{3.23}$$

where M is the delay in samples and b_k are the long-term prediction coefficients. The value of M corresponds to the number of samples of a delay in the range from 2.5 to 18 msec . This corresponds to a pitch period for voiced speech segments. The delay is random for non-voiced speech. The delay range corresponds to a pitch range from 56 to 400 Hz which covers variation in pitch for a wide variety of speakers.

In this implementation, the pitch delay ranges from 20 to 147 every odd subframe while even subframes are search and coded within 32 lags relative to the previous subframe. The delta search greatly reduces computational complexity while causing no perceivable loss in speech quality. This adaptive code book search is carried out four times per frame (every 7.5 ms). The search is performed by closed-loop analysis using modified minimum squared prediction error (MSPE) criteria of the perceptually weighted error signal. The gain and index are coded using absolute, non-uniform, scalar 5 bit quantization as specified in the standard 1016.

The MSPE search criteria is modified to check the match score at submultiples of the delay to determine if it is within 1/2 dB of the MSPE. The shortest submultiple delay is selected if its match score satisfies our modified criteria. While maintaining high quality speech, this results in a smooth "pitch" delay contour that is crucial to delta coding and the receiver's smoother output in the presence

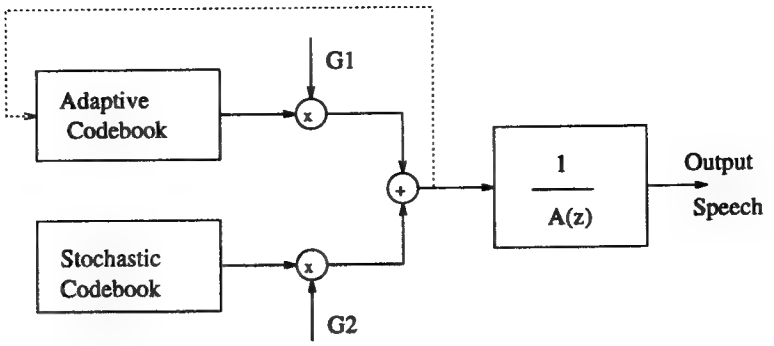

Figure 3-3: The two codebook CELP synthesizer model

of bit errors.

3.4.5 Excitation

The excitation parameters (codebook vectors) are determined for a frame of samples such that the (weighted) mean squared error over the same frame is minimized. By using an adaptive codebook representation, the input to the short-term synthesis filter is a linear combination of both adaptive and fixed codebooks. A block diagram is shown in Figure 3.3. The codebooks in Figure 3.3 are searched sequentially; that is, the adaptive codebook is searched first and then the stochastic codebook is searched.

The consistency introduced by the adaptive codebook during voiced segments of the input speech is perceptually important. So the relative adaptive codebook component is increased in voiced regions and stochastic codebook component is slightly reduced. This is done after the adaptive codebook has been searched as explained in the previous section. The efficiency of the adaptive codebook is determined by computing the cross- correlation of the excitation before and after pitch prediction. If the efficiency is greater than 0.9, the stochastic codebook component is amplified only slightly.

The stochastic codebook search is performed by closed-loop analysis using con-

ventional minimum squared prediction error criteria of the perceptually weighted error signal. The code book is overlapped to allow recursive computational savings. The codebook is 77% sparse (zero-valued entries) and this is also used to minimize the computations. This codebook has been found to cause no degradation in speech quality relative to other types of codebooks and significantly reduces search computation.

3.4.6 Postfiltering

Postfiltering reduces the perceptual coding noise. The general idea is to emphasize the spectral peaks in the speech signal predicted by the short-term LPC analysis. In noise spectral shaping, lowering noise spectral components at certain frequencies can only be achieved at the price of increased noise components at other frequencies [11]. At low bit rates (4800 bps), the average noise level is quite high. It will be very difficult to force noise below the masking threshold at all frequencies. So a better idea would be to preserve the formant information by keeping the noise as low as possible in the formant region. Now the noise will be above the masking threshold in the valley region. This can be attenuated by using a postfilter. But this postfilter also attenuates the spectral components in the spectral valleys. Fortunately, the *just noticeable difference (JND)* for spectral valleys can be very high and so attenuating the spectral components in the spectral valleys introduces only minimal distortion.

[12] uses such a postfilter. For an all-pole LPC synthesis filter $1/[1 - P(z)]$, the corresponding postfilter is $1/[1 - P(\alpha z)]$, where $0 < \alpha < 1$. This postfilter reduces the perceived noise level but results in muffling of the speech. This is due to the low-pass spectral tilt in the response of the filter for voiced speech as shown in Figure 3.5(?). This spectral tilt is reduced by adding zeros having the same phase angle as the poles but will smaller radii. This has the effect of subtracting the responses of two filters, one with both formant peaks and the spectral tilt and

the other with only the spectral tilt. The resulting filter transfer function has the form:

$$H(z) = \frac{1 - P(z/\beta)}{1 - P(z/\alpha)}, 0 < \beta < \alpha < 1. \tag{3.24}$$

Since some amount of muffling was still observed, a first order filter which provides a slightly high-pass spectral tilt was added. This filter has a transfer function of $[1 - \mu x^{-1}$, where μ is typically 0.5. So the complete transfer function of the postfilter looks like:

$$H(z) = \frac{[1 - P(z/\beta)][1 - \mu x^{-1}]}{1 - P(z/\alpha)}, 0 < \beta < \alpha < 1. \tag{3.25}$$

An automatic gain control (AGC) is added to the postfilter to avoid occasional large gain excursions. The AGC ensures that the filtered speech has roughly the same power as the unfiltered speech. This is done by estimating the power of the speech before and after filtering and using the ratio of the two values to scale the output speech.

Table 3.1 provides a summary of the CELP algorithm used.

3.4.7 Performance of the CELP coders

CELP coders do not exhibit the usual vocoder problems in background noise because they use a more sophisticated model than the classical vocoder's pitch and voicing (eg., LPC-10). Background noise including multiple speakers has been found to be faithfully reproduced [8]. Speech intelligibility and quality have been measured using the Diagnostic Acceptability Measure (DAM). [8] reports the DAM scores of the 4800 bps CELP coder in different environments. It gives a score of 65 in a quiet environment and 58 in an office environment (the input

Table 3.1: 4800 bps CELP characteristics

Property	*Spectrum*	*Adaptive CB*	*Fixed CB*
Update	30 ms	7.5 ms	7.5 ms
Parameters	10 LSPs (independent)	1 gain, 1 delay 256 codewords	1 gain, 1 index 512 codewords
Analysis	Open loop 10 th order autocorrelation 30 ms Hamming no preemphasis 15 Hz BW exp	closed loop size 60 mod MSPE VQ weighting =0.8 delta search range: 20 to 147	closed loop size 60 MSPE VQ weighting =0.8 shift by -2 77 % sparsity ternary samples
Bits per Frame	34 (3444433333)	index: 8+6+8+6, gain: 5x4	index: 9x4 gain: 5x4
Rate	1133.33 bps	1600 bps	1866.67 bps
Miscellaneous	The remaining 200 bps are used as follows: 1 bit per frame for synchronization, 4 bits per frame for forward error correction and 1 bit per frame for future expansion		

speech has a DAM of 84).

Chapter 4

Optimization Techniques

For real-time operation, all the operations on one frame need to be performed within one frame interval (30 msec). This calls for algorithm level and code level optimizations. The CELP algorithm is computationally complex and intensive research over the years has lead to real- time implementations.

4.1 Algorithm Optimization

The algorithm optimizations incorporated are:

- Use of two codebooks, adaptive and stochastic. The excitation is determined for short frames. Since long term prediction used previous excitation signals, the selected excitation signals and long term coefficients affect future choices. So for each frame a few best choices are maintained and in the next frame, different excitation signals are selected given different initial conditions.

- Overlapped stochastic codebook. Each codeword is 2(M-index) to 2(M-index)+L-1 , where M is the maximum code book size which is 512 and L is the code word length which is 60. Codewords are overlapped by shifts of 2 along the code vector.

- Codebook sparsity. The codebook is 77% sparse, so the computations for the zero elements need not be carried out which results in large savings in the computations.

4.2 Code Optimization

In addition to algorithm optimization, the underlying code also has been optimized. The TI C compiler version 4.6 [15] was used along with the in-built optimizer for generating compact code. This optimizer used at level 3 has features such as simplifying loops, rearranging statements and expressions and allocating frequently used variables into registers. It also expands calls to small functions inline and converts array references in loops to incremented pointer form.

Unfortunately the present version of the TI C compiler does not recognize DSP specific functions like convolution and filtering which can be implemented more efficiently using circular buffers. Therefore filter routines were hand-coded using circular buffer implementations [16] which resulted in computational savings. For example, a compiler generated optimized filter routine that took 10600 cycles to execute took only 4200 cycles using a circular buffer implementation.

Sometimes only a portion of the code has DSP specific routines like convolution and the rest of the code has already been optimized by the compiler. In these cases, the DSP specific routines were inlined into the compiler generated assembly code.

4.3 SPW Implementation

The encoder and the decoder were implemented as SPW blocks. At present, these blocks are part of the SPW library. A user can pull down these blocks and construct a system which can then be run the TMS320C40 processors on the DBV4x board.

Since the encoder and the decoder are usually on the transmitter and receiver portions of a system (physically separate), the encoder and the decoder were implemented on different processors. The configuration of the LSI board is such that the A/D board can be accessed only via processor 1, this processor was used to buffer the input data and then send the output from the decoder to the speaker.

The blocks for the encoder and the decoder were developed on SPW (the software tool developed by the Alta Group of Cadence) [14] using the development environment available for Custom-Coded Blocks. The following paragraphs provide some detail on writing these blocks.

During the creation of any Custom-coded block, SPW generates an expression file with an EXPR extension. This file has sections for include files, include directories, link options, input variable declarations, output variable declarations, state declarations (for local and global variables), initialization code, run-time code and termination code. The input and output variables are filled in by SPW during the generation of the expression file. The other sections are initially blank and they need to be filled in by the user. The initialization code for all the blocks is run once before the actual iterations start. The run-out code is run for every iteration and the termination code is run when all the iterations are over. The number of iterations desired is a parameter that can be specified during run-time.

When the user builds a system, SPW generates the code for each of the blocks. The dsp-shell of SPW then compiles and makes an executable code. This code is then downloaded onto the DSP processor and run. In the original version of SPW, the code compiled for the C40 was not able to be linked to the libraries specified in the link options. So the dsp-shell was changed to accept the link options as a variable.

The expression file was written in C. This file consists of the top level functions of the encoder. These functions in turn call other functions and so creates a nest of function calls. The object files for all these functions are part of a library and these

functions get linked when the linker is run. The object files were initially generated using the Code Generation System (CGS) on SPW. But it was found that the code generated was not optimal in terms of the usage of circular buffers, loops, etc. So the functions were hand-written in the C40 assembly language. Then the object files were created using TI's assembler and the files thus generated were archived into a library. This ensured that the code could run real-time while at the same time remaining within the SPW block-diagram environment. This arrangement also cut down the development time for the blocks. The usual practice would be to simulate the algorithm on SPW and then go in for DSP processor specific assembly language programming. But in the current set-up, one can directly program in assembly language, link the object file to the library and immediately run the system from SPW and check the output. This set-up significantly reduced the algorithm development time.

Chapter 5

Multiprox and DSP

There are two objectives of this Chapter which are motivated by the need to build and simulate real-time systems on a DSP network of processors in a rapid prototyping environment. Our choice for the DSP is the TMS320C40 processor which is customized for DSP operations in a distributed environment.

The first task is to accomplish a reduction in the inter-processor communications overhead while using Multiprox with the C40. Improvements in speed of the order of 30 to 40, have been observed with our modifications. With our ipc handling code, most of the overhead is in the initialization phase of the DSP code and there are less than 20 assembly level instructions that are executed during each iteration of the actual run phase. Furthermore, our techniques can be used with buffered transfers at no additional cost, i.e. transferring say 1000 words takes about a total of 25 instructions, while a single word transfer still takes 20 instructions.

The second task is to write highly efficient DSP code for the blocks in SPW (Signal Processing Worksystem, Alta Group). Currently the block codes are general purpose C program sections. There are a number of possible modifications that can lead to efficient DSP programs for systems built using SPW.

The rest of this Chapter is organized as follows. In the first Section, an intro-

duction to Multiprox and the routine steps followed in running the software is provided. In Section 2 an outline of the actual interprocessor communication functions (low level) used by standard Multiprox is provided. The reasons for the poor efficiency of the algorithms when applied in low complexity systems are also mentioned. Section 3 details the two alternate algorithms suggested by us while Section 4 compares the performance of our methods with the standard Multiprox and single C40 and brings out the advantage of using our techniques. The second task is discussed in Section 5.

5.1 Standard Multiprox

Multiprox is an optional software product from the Alta Group (formerly Comdisco Systems Inc.) that works seamlessly with SPW to provide a distributed processing environment. Though several types of processors are supported by Multiprox, only parallel processing with the C40 will be investigated. The C40 is very well suited for such applications since it was designed with distributed processing in mind. One should expect to see a very good utilization of the processors in most distributed algorithms since the interprocessor communication(ipc) overhead between C40s can be made very small. However this is not the case with Multiprox and in many cases opting to run the code in parallel on several C40s can actually slow down the throughput compared to a single C40. For the class of low to medium complexity algorithms we observed anywhere from 20 times slower to no speed up at all. Clearly this is not what the designers of Multiprox had in mind and it could initially be frustrating. The problem lies with the way Multiprox does the ipc for the C40 processors. We suggest two alternate techniques for handling the ipc. Both the methods yield speed-ups (compared to a single C40) that are unparalleled by any other method.

Standard Multiprox is a multiprocessor code development software that allows

the user to run a block oriented system in SPW, on several processors. The processors used have to be of the same kind. For example one cannot use a C40 along with a SPARC processor. One can put the software in perspective and its interactions with the other software products of SPW as shown in Figure 1.

Once a block diagram has been created, the user has to launch a Multiprox editor from the block diagram editor. A partition of the designed system needs to be specified. For optimum performance, the partition has to be carefully selected so that the ipc per iteration is minimized and at the same time the processors are more or less equally loaded. This requires some practice and many times it also could be time consuming. Many block diagrams simply cannot be partitioned so that the code is equally distributed among all the processors. In any case, once a satisfactory partition has been achieved, the user has to edit the "region parameters" file for each partition. The file consists of the "processor value" and "processor type" which target the specific processor (in a multiprocessor environment) and the type of the processor. Typically for C40s, the first parameter takes numbers 1, 2, 3, 4 and so on while the processor type would be C40. The exact details of this depend on the software that actually interacts with the processors (mostly designed by the board vendors). After these parameters have been correctly specified, the user needs to specify an "Architecture Description File" (ADF for short) which contains the description of the multiprocessor set up and also specifies exactly which ipc blocks are to be used for interprocessor communications. After verifying (or creating) the ADF file, the software is ready to create, compile and run the partitions on the respective processors.

5.2 IPC in Standard Multiprox

Standard Multiprox (abbreviated to S-MPX) inserts an input and an output block at each connection between partitions. These blocks (which are hidden from the

block diagram) are used to specify the exact ipc method.

As an initialization step, S-MPX establishes logical (software) links between processors using four functions, *fifoInitX()* and *fifoConnectX()* where *X* could be *Input* or *Output*. The *fifoInit* functions are called during the initialization stage. The functions establish the actual physical connections that exist and also set up the software or logical links for each signal that runs across a partition. They also do an initial synchronization by a fairly complicated procedure of exchanging tokens, the details of which may be found in the *CGS/Multiprox Interface Porting Kit for C40* that is distributed by the Alta Group of Cadence Systems Inc. The *fifoConnect* functions are used to synchronize the processors at the beginning of the run stage. This level of synchronization assures that all the processors are actually at the beginning of the run stage. All of these functions are of major complexity and typically take an order of 10^5 cycles. The actual ipc occurs via a call to either *fifoRead()* or *fifoWrite()* and is analyzed below.

Assume that processor A has a 32 bit word to transfer to processor B. An autoinitialization sequence (see Section 3) of 7 words is first sent to processor B whose corresponding DMA channel uses these to autoinitialize itself (without any CPU intervention). After this set of transfers is successful, the DMA receives the data from A through the pair of connected comm. ports. CPU B then attempts to send an autoinitialization sequence for the DMA counterpart on A. This involves transferring another set of 7 words which are used by the DMA of A to autoinitialize itself. On top of these, there is also a token exchanged with each set of transfers. Thus there is a net transfer of 17 words (32 bits each) between the two processors for each word of data. In addition to the 68 cycles that it takes to transfer these 17 words, there is a large overhead of function calls and branch statements. Consequently each 32 bit data transfer operation should take about 150 instruction cycles (each cycle being 40 or 50 nanoseconds). However on an average, on the order of 600 to 800 cycles are consumed instead. The cause of this

extraordinarily large number of cycles is explained below.

When the SPW timing macros are used (SET_TIMER and GET_TIMER macros defined in *cgsio.h*), the measured value of the time spent for the ipc. in S-MPX is always about 140 to 150 cycles. Yet on an average, 600 to 800 cycles are spent for the ipc functions ! Similar to the uncertainty principle, the act of making the measurement slows down the processors enough that there is no anomalous behavior (the increased time spent per iteration is due to the printf statement in the GET_TIMER macro that needs to write the measured time on the host SPARCstation). On the other hand if the measured time is written only once in a while, say whenever it crosses 10000 cycles, then occasionally some high time intervals are recorded. Tables 1, 2 and Figures 2, 3 show these unusual time intervals while executing "mpxmit2.system" (shown in Figure 4) or "mpxfgen0.system" (shown in Figure 12, also called SYSTEM-1 elsewhere). The number of iterations and the threshold above which the times are recorded are also listed. The entire simulation time (in cycles) is also a random variable as shown in Table 3 !

	Unusual Time Intervals in Cycles
mpxmit2.system 7000 iterations Threshold = 1000	278408, 47664, 46846, 40078, 25818, 27208, 25024, 48810, 26002, 51480, 29138, 28178
mpxmit2.system 7000 iterations Threshold = 30000	270048, 40174, 40146, 43686, 41470, 50080, 46000, 43716, 45150, 70314, 41794, 41746
mpxmit2.system 25000 iterations Threshold = 60000	271872, 165424, 70612, 185424
mpxmit2.system 500000 iterations Threshold = 2000000	2686442, 6321058, 6510172, 2149998, 12825392, 15671034, 12475222, 6544504, 6647518, 6642282
mpxmit2.system 500000 iterations Threshold = 10^7	14715414, 17935192, 13144598, 15177082 13901878, 12009398, 13149688

Table 1: Unusually high time intervals for mpxmit2 system

	Unusual Time Intervals in Cycles
mpxfgen0.system 10000 iterations Threshold = 10000	279080, 13780, 251650, 51772, 178302
mpxfgen0.system 7000 iterations Threshold = 30000	267486, 53188, 45456, 2373836, 30876, 117934
mpxfgen0.system 25000 iterations Threshold = 60000	268300, 313648, 652516, 999202, 1111570, 340144, 359946, 184486, 193048, 766716
mpxfgen0.system 500000 iterations Threshold = 10^7	274906, 128640, 177950, 115960, 981606, 120918, 1219390, 281420, 130190

Table 2: Unusually high time intervals for mpxfgen0 system

Iterations	Time (cycles)	cycles/iteration
30000	16918044	564
30000	16148050	538
30000	15982120	533
30000	17989946	600
30000	16452756	548
50000	39746014	795

Table 3: mpxmit2 simulation time as a random variable

There is an unavoidable possibility of "deadlock" between the C40 processor and the corresponding DMA channel, which involves the peripheral bus and the comm port. The autoinitialization of the DMA occurs without any synchronization with the comm port. The CPU write operations also proceed with no synchronization. Consider the following situation (also illustrated in Figure 5).

The DMA of processor A (DMA_A) tries to autoinitialize from the comm port (this situation occurs immediately after every data transfer). Suppose that the CPU of A (CPU_A) tries to write the autoinitialization sequence for the DMA of B (DMA_B). Similarly the CPU of B (CPU_B) tries to write to its comm port. Such a scenario leads to deadlock and the peripheral bus hangs on both A and B, and so does CPU_A, CPU_B, DMA_A and DMA_B. The problem however seems to go away after a large number (typically 10^5) of cycles. Hence it is technically a resource starvation problem rather than a deadlock situation. Nevertheless, it is unclear as to why the problem gets resolved or even if it is the main cause of the anomalous behavior. A few postings made to the internet newsgroup "comp.dsp" have elicited interesting suggestions ranging from faulty comm port design of the C40 to noise on the links.

The efficiency of S-MPX ipc goes down even more dramatically if more than one logical link exists between any two partitions. Granted that in any multiprocessor algorithm, the overall execution can only proceed as fast as the slowest (most loaded) processor, the performance of S-MPX when more than link exists is simply unacceptable in any real-time system. The results listed in Table 4 verify this statement. The reason for this also seems to be the occurrence of "deadlocks", which will happen more frequently because of the increase in the number of ipc. transfers.

5.3 Our IPC Methods

The key issue is how to handle the ipc so as to achieve an acceptable utilization from multiple processors and to avoid the problems of S-MPX. For simulation purposes, if a buffer is provided so that the ipc functions are executed for blocks of data, there is a significant improvement in the CPU utilization even for S-MPX. On the C40, the speed of execution can also be affected if one provides a buffer. In order to achieve a good throughput, the overhead of the ipc should be distributed over the entire buffer length. In a real time implementation, this is important because, suppose the ipc transfer of the buffer is done between any two samples (or iterations) of the system. Unless the A/D can interrupt the transfer process, samples will be lost at the A/D. Interrupt driven I/O is computationally costly owing to the latency of the processor, function call overheads, etc. On the C40, the distribution of the ipc can be readily achieved using the DMA operations. An efficient ipc protocol was developed and is given in Section 1.3.2. In the following Section the advantages that the TMS320C40 processor provides are summarized.

5.3.1 The TMS320C40 Processor

The TMS320C40 is highly optimized for parallel processing applications.

Some of the advantages of using the TMS320C40 as a parallel processing node are

- Node speed of about 50 MFLOPS.
- Single cycle (50 nanoseconds) 32 bit floating point multiplier.
- Single cycle (hardware) divide and inverse square root.
- Single cycle floating point multiply and add (parallel operations).
- Single cycle delayed branch loops.
- Six 20 Mbytes/sec comm ports for node to node communications.
- Six bidirectional DMA channels for ipc.
- Highly configurable for parallel processing topologies.
- I/O of 100 Mbytes/sec on Global bus, 100 Mbytes/sec on the local bus and 120 Mbytes/sec on the six comm ports.

5.3.2 Parallel Processing with C40

Some of the features of the C40 such as dedicated parallel communications ports (commport) and DMA channels allow us to develop an efficient protocol. Each commport is bidirectional and can be used to send and receive data simultaneously. Port arbitration is done entirely by the C40 hardware and thus it is much simpler to transfer data in both directions. The total commport transfer rate of 20 MBytes per second allows for a high degree of parallel processing. One ipc algorithm that can be proposed is based on a polling scheme where the CPU polls the appropriate DMA control register when it is ready to transmit or receive data as shown in Figure X. If the previous DMA transfer (to or from the relevant commport) was completed then the CPU writes or reads from the DMA buffer and restarts the DMA for the transfer operation. This procedure decouples the CPU from doing the memory transfers completely and also prevents the peripheral bus from getting halted. However there are exceptional cases where the appropriate flags in the DMA control register are not properly set. We observed that the

combination of all the four methods (mentioned in Chapter 9 of the C40 manual) is always OK to use.

The Scalar IPC (SIPC) method is very similar except that it does not rely on the DMA control register (see Figure 8). In this case, the receiving processor automatically generates an acknowledgement that is transferred to a known memory location on the transmitting processor. All this handshaking is done by the respective DMAs and the CPU is not bothered at all. When the C40 needs to send or receive data, it checks the memory location for an acknowledgement.

The major advantage of SIPC is that in systems with feedback between the two processors, the feedback signal can very easily be piggybacked on the acknowledgement link. As shown in the results the penalty for an extra data transfer is 2 cycles (or 100 nanoseconds) per iteration which is truly negligible.

In most signal processing applications, data transfers between the processors occur in a deterministic order and the transfers are done periodically. We make effective use of this simplicity by constructing a linked list of DMA autoinitialization sequences as shown in Figure 9. Each linked list element consists of seven fields (clarity permits only a couple of relevant ones to be shown). The head and tail of the linked list are used to signify a halt code which enables any C40 in the network to halt all the other processors if necessary. The rest of the elements of the list contain the addresses of the data and direction of data transfer. In Figure 9, the direction field is a 0 if data is transmitted and is a 1 if data is received. Further simplification can be achieved in systems where data transfer occurs in one direction only. In that case one can buffer all the data to be transferred in each iteration and do the transfer operation only once per iteration.

As listed in Table 4, SIPC takes 24 cycles or 1200 nanoseconds for transferring one word (4 bytes). There is a case of "diminishing returns" when one uses more processors. Since a single word read/write takes about 24 cycles, even in

the hypothetical case where each sample (or iteration) takes the same number of cycles on each C40, the algorithm complexity cannot be less than 24 cycles.

The complex IPC blocks *srinivas/cipc_in* and *srinivas/cipc_out*

In systems that can withstand delays (eg. those that do not have feedbacks), all ipc methods benefit to varying extents by the use of buffered transfers. The basic method is illustrated in Figure 10 for the cases of S-MPX and either of the methods suggested by us. For FIPC, if the DMA is transferring the contents of buffer B_1 then at the same time, the CPU can write to another buffer B_2. When B_2 becomes full, the CPU can poll the DMA channel's control registers to determine if B_1 was successfully transferred. Then the buffers need to be switched and the processors can continue. Whereas by switching buffer pointers (which takes a couple of cycles), our methods can essentially transfer any size buffer in about 24 cycles, there is no advantage of using two buffers and switching them while using S-MPX. This is because unlike our techniques, S-MPX does not operate the DMA in parallel with the CPU. Of course we have to assume that the DMA finishes the previous transfer before the CPU is ready with the new buffer in order to realize the entire buffer transfer in 24 cycles or less using our methods. In most DSP type algorithms this will be true since the CPU will do a certain amount of processing before it is ready with the buffer.

There is also the issue of CPU and DMA conflicts when they try to access the same resources. When there is only a scalar word (4 bytes) being transferred, the maximum probability of conflict occurs when the DMA channel is autoinitializing because it involves 7 word transfers in the unified mode (see the TMS320C40 Manual for more details). Thus it is 7 times more likely to clash with the CPU when it is in its autoinitialization procedure. On the other hand for buffered transfers, CPU/DMA conflicts can occur more frequently and depending on the size of the buffer it might not be as important to consider the effect of the autoinitialization sequence itself.

One important feature of the C40 that is relevant here is the concurrent operation of the DMA processor without almost any intervention from the CPU. Thus if the DMA is transferring the contents of buffer B_1 then at the same time, the CPU

can write to another buffer B_2. When B_2 becomes full, the CPU can poll the DMA channel's control registers to determine if B_1 was successfully transferred. Then the buffers need to be switched and the processors can continue.

The results of buffered transfers in S-MPX is listed in Table 6. The system used to get the timing results was of high complexity (a few thousand cycles per iteration). Thus running a complicated algorithm on two processors using S-MPX did produce a speed up of nearly 2.

The blockcodes of the IPC blocks are similar for all the types. The autoinitialization sequences of the DMA are first set up for all possible operations of the particular DMA channel. This is done using the "autoinit" link structure. The DMA channel index itself is obtained from a connectivity matrix that is included in the "mytypes.h" include file (see page xx for the details of the mytypes.h). The values of the different fields of the linked list element are set up. The number of data words to be transferred depends on the number of signals that are input or output from the block. CPU accesses are given priority over DMA accesses by setting the corresponding DMA channel control register bits.

In SIPC, the control register is also set for a write operation to the comm. port. Following the write, the DMA channel automatically autoinitializes for receiving data or acknowledgement. The write operation involves either data to be transferred (if the processor is the transmitter) or the acknowledgement corresponding to *the data received on the previous transfer* if the processor is the receiver. After the read operation (either data or acknowledgement) the DMA channel halts and waits for the CPU to restart it. For its part, the transmitting CPU first checks if the previous transfer has been acknowledged. If so it restarts the DMA for the next transfer. On the other hand, the receiving CPU checks its OK flag (the last word of the transferred data buffer). If the flag is set then it reads from the data buffer and restarts its DMA channel. In this manner the CPU is almost completely free of involvement in the memory transfers.

5.4 Comparison of the Methods

The CPU utilization of a processor, defined as the percentage of time that the CPU actually executes the user's program, not including the interprocessor communication (ipc) overhead, is fairly good when Multiprox is used for systems that are computationally intensive. However simpler systems that take of the order of a few hundred instruction cycles do not benefit from the use of Multiprox. Since typically rapid prototyping involves quick (real time) design prototyping which in many cases would also be reasonably of low complexity (a highly complex system is simply not suitable for rapid prototyping, at least using limited resources !) multiprocessor routines have to work with similar low level throughputs, i.e. low complexity and yet large volume data transferring capability is essential for any interprocessor communication system.

We compared the performance of all the ipc methods suggested in this Chapter with that of using a single C40. The system that was chosen for this purpose is shown in Figure 11. SYSTEM-1 and SYSTEM-2 that were used to obtain the results in Table 4 are shown in Figures 12 and 13 respectively. The difference in the two systems was that the number of words transferred per iteration in SYSTEM-1 was 1 while it was two word transfers per iteration for SYSTEM-2. We also checked the real time capability of our algorithms against S-MPX and a single C40 (using CGS). Specifically we modeled the Delta Modulation/Demodulation system using the different ipc algorithms and CGS (on a single C40). The results are given in Table 5. It is clear that our methods prove advantageous at high data rates (due to the low complexity of the ipc algorithms).

All the Multiprox algorithms were run on two processors. Also the test for real time capability used in Table 5 is that the rate of iterations (or rate of sampling) is the same as the sampling frequency set for the D/A conversion on the first processor. If the rate is slower than the sampling frequency, then the system is not real

time.

System	Method used	IPC	Time
SYSTEM-1	SIPC	2	54 sec
SYSTEM-1	Standard mpx	2	181 sec
SYSTEM-1	Single C40	2	99 sec
SYSTEM-2	SIPC	1	54 sec
SYSTEM-2	Standard mpx	1	84 sec
SYSTEM-2	Single C40	1	98 sec

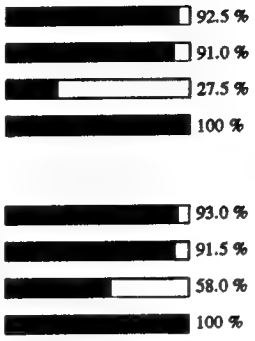

Table 4: Timing summary of SYSTEM-1 and SYSTEM-2

Tone in	samp.Freq	Method used	IPC	Time	Real-time
7 kHz	67.4 kHz	SIPC	2	45 sec	yes
7 kHz	67.4 kHz	Standard mpx	2	174 sec	no
7 kHz	67.4 kHz	Single C40	2	99 sec	no
3 kHz	29.6 kHz	SIPC	2	101 sec	yes
3 kHz	29.6 kHz	Standard mpx	2	160 sec	no
3 kHz	29.6 kHz	Single C40	2	101 sec	yes
1 kHz	9636 Hz	SIPC	2	311 sec	yes
1 kHz	9636 Hz	Standard mpx	2	311 sec	yes
1 kHz	9636 Hz	Single C40	2	311 sec	yes

Table 5: Timing summary of the system in Figure 11

We also benchmarked buffered S-MPX and CGS on the SPARC on two other systems that had almost loaded both the processors equally. The results are presented in Table 6. The systems differed in the computational complexity. SYSTEM-1 was much more complex than SYSTEM-2. The results given are for 10^6 samples. The details of the systems used are not important here and thus are not given.

SYSTEM	Method used	Buffer Size	Time in secs
SYSTEM-1	SPARCstation II		66.0
SYSTEM-1	Single C40		35.0
SYSTEM-1	Multiprox on 2 C40s	1	19.0
SYSTEM-1	Multiprox on 2 C40s	1000	17.0
SYSTEM-2	SPARCstation II		22.0
SYSTEM-2	Single C40		28.0
SYSTEM-2	Multiprox on 2 C40s	1	32.0
SYSTEM-2	Multiprox on 2 C40s	1000	17.0

Table 6: Timing summary of SYSTEM-1 and SYSTEM-2

While the SPARCstation was executing the systems, about 90 to 95% of the CPU was used by the programs. One can deduce that using a large enough buffer with SPW Multiprox, there is very good processor utilization (if the computations are sufficiently intensive). However as mentioned earlier, this cannot be used in a real time system. Also the SPARCstation outperforms the C40 when the complexity of the algorithm is very low. Typically most of the operations done in SYSTEM-2 were of the logical kind and thus the reduced architecture of the SPARCstation

II processor provides advantages in such cases. However the C40 DSP processor outperforms the SPARC processor significantly when the algorithm is fairly complex, involving floating point operations.

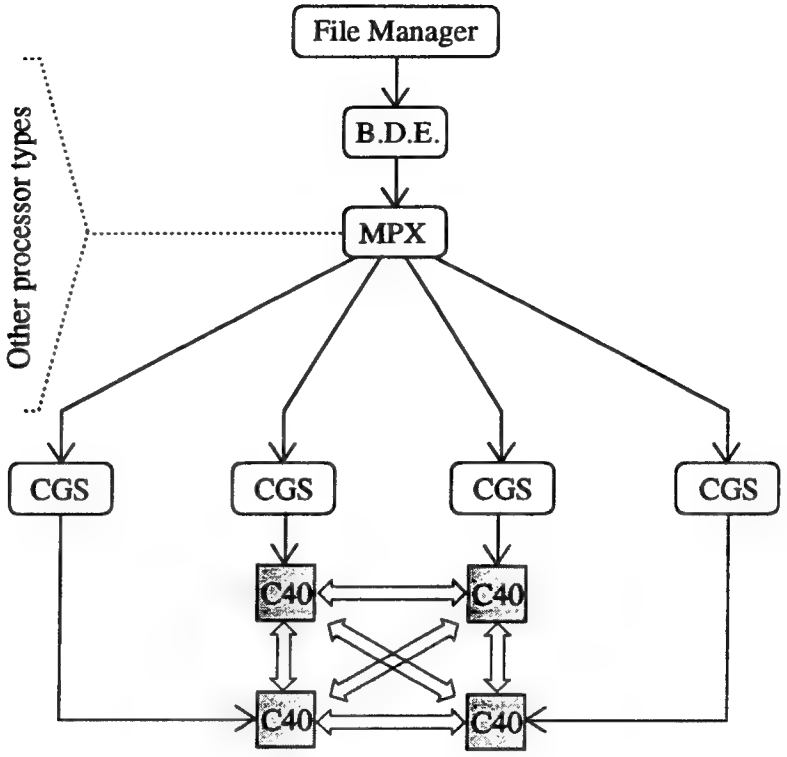

Figure 1: Multiprox and its interactions with the rest of SPW

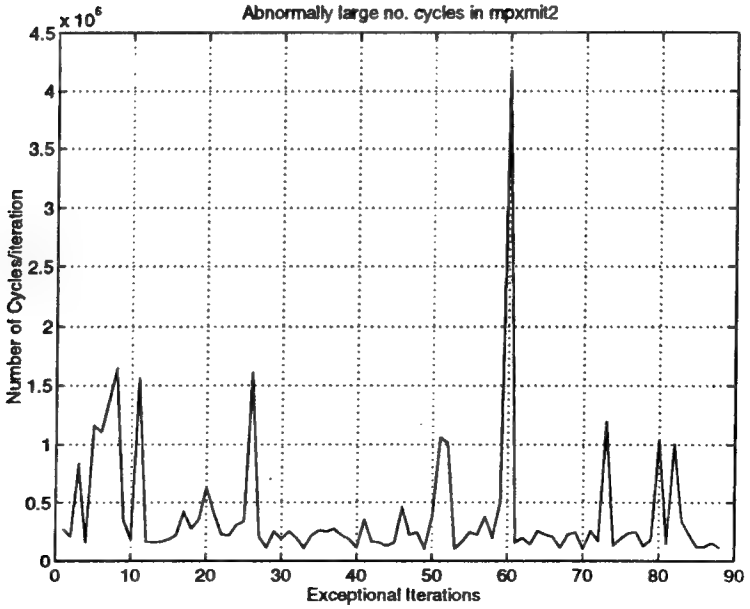

Figure 2: Abnormal Iterations in the Simulation of "mpxmit2"

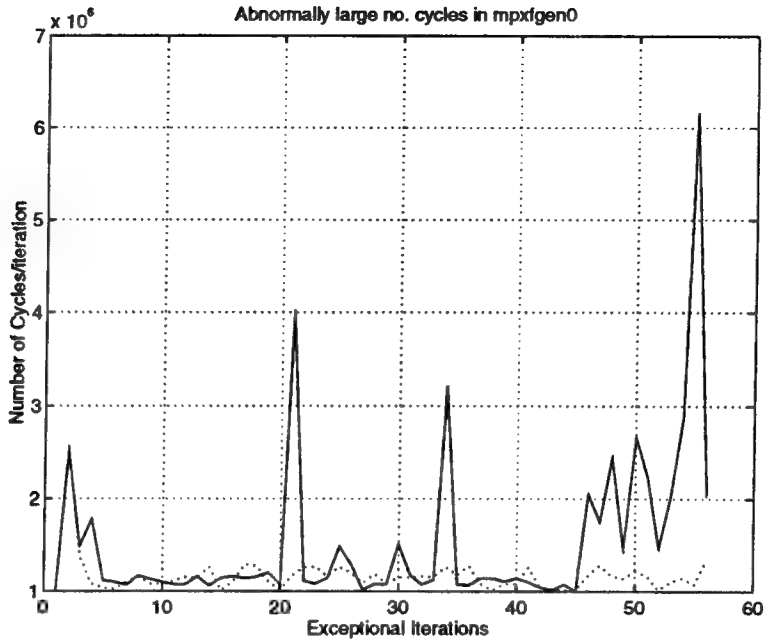

Figure 3: Abnormal Iterations (2 traces shown) of "mpxfgen0"

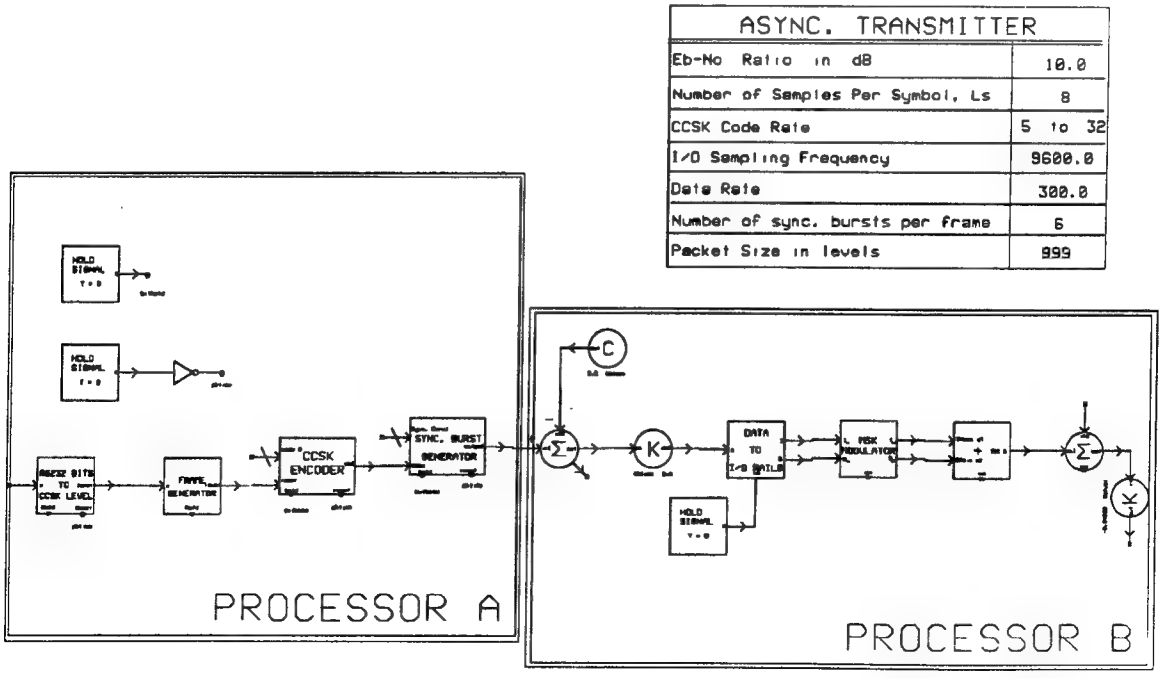

Figure 4: SPW System: "mpxmit2.system"

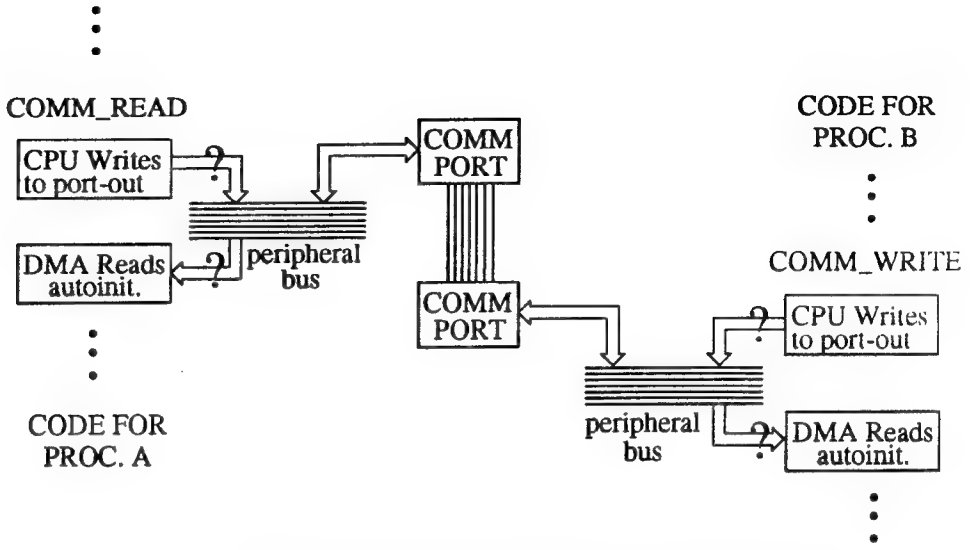

Figure 5: Deadlock in standard Multiprox

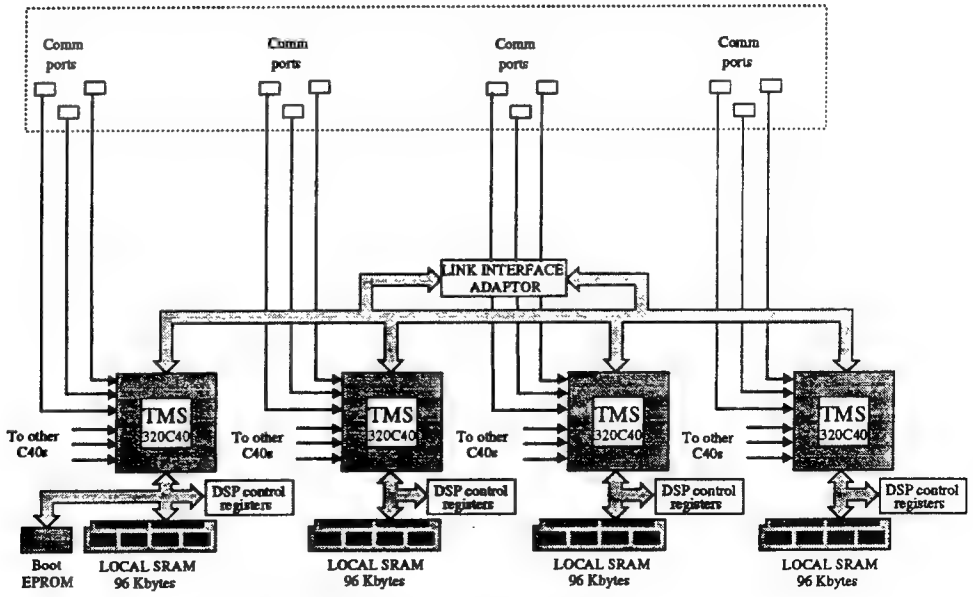

Figure 6: Quad C40 Board Layout

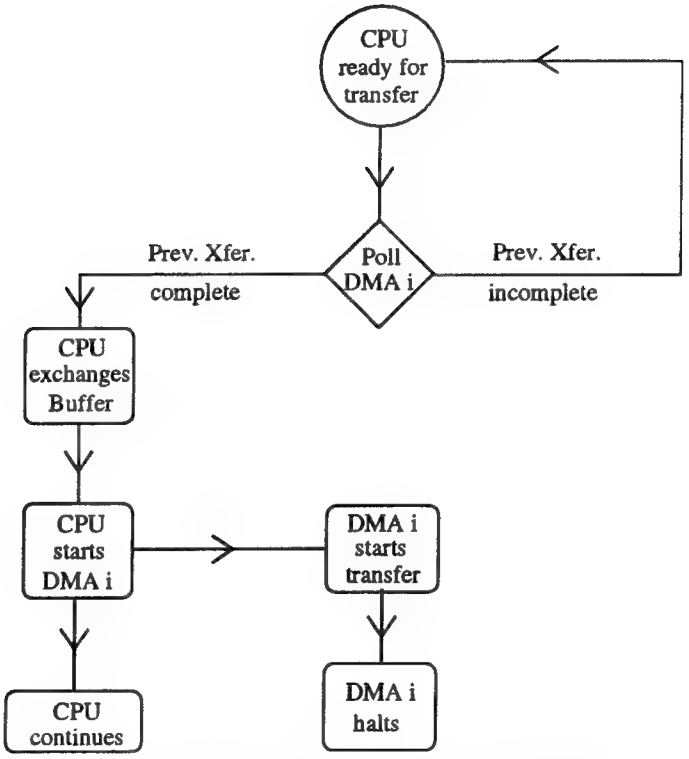

Figure 7: CPU-DMA Interactions for SIPC

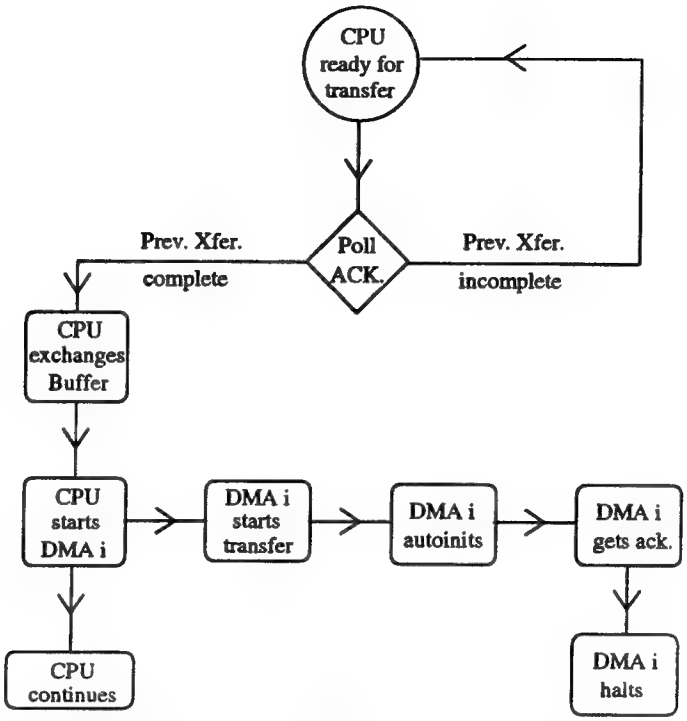

Figure 8: CPU-DMA Interactions for SIPC

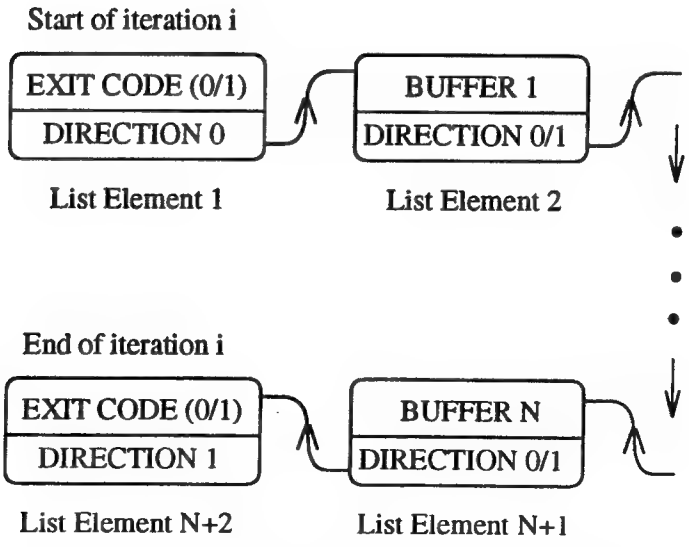

Figure 9: Linked List & Operation of DMA

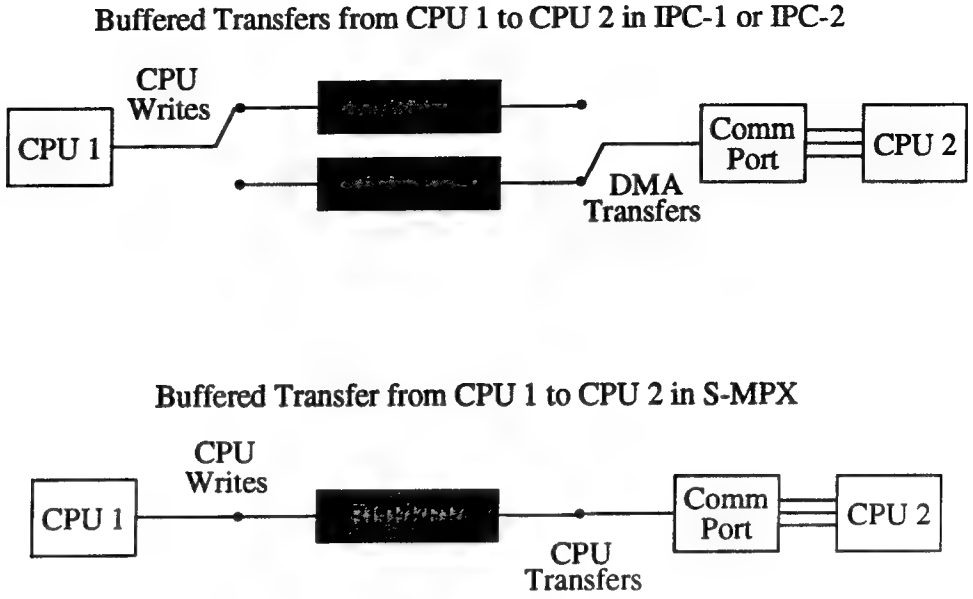

Figure 10: Use of buffers for SIPC and S-MPX

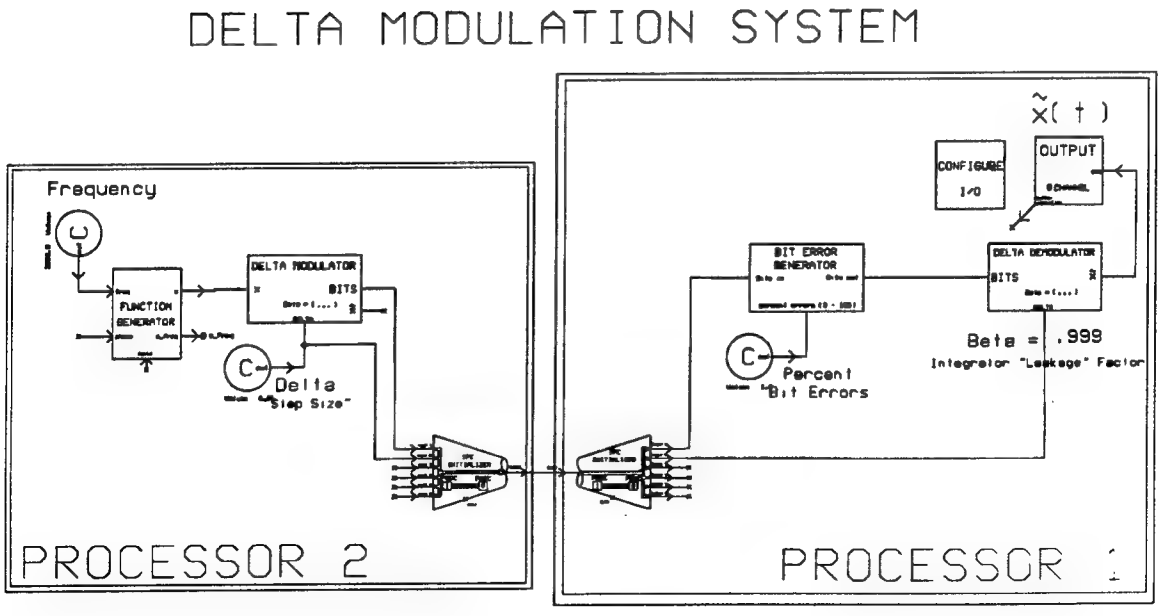

Figure 11: Delta Modulation example

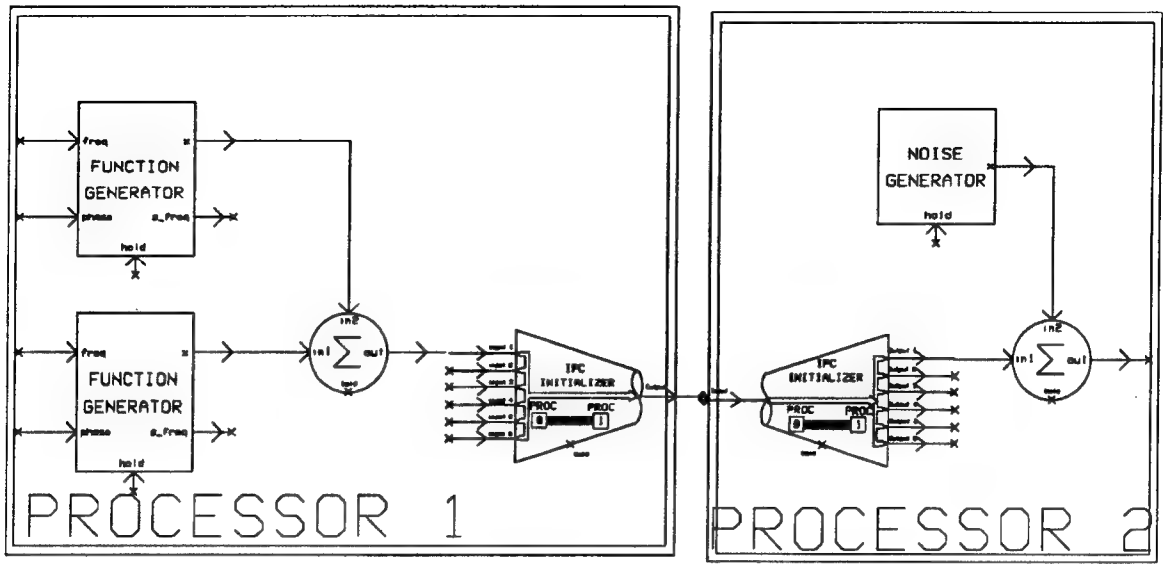

Figure 12: SYSTEM-1 with one data transfer used in Table 4

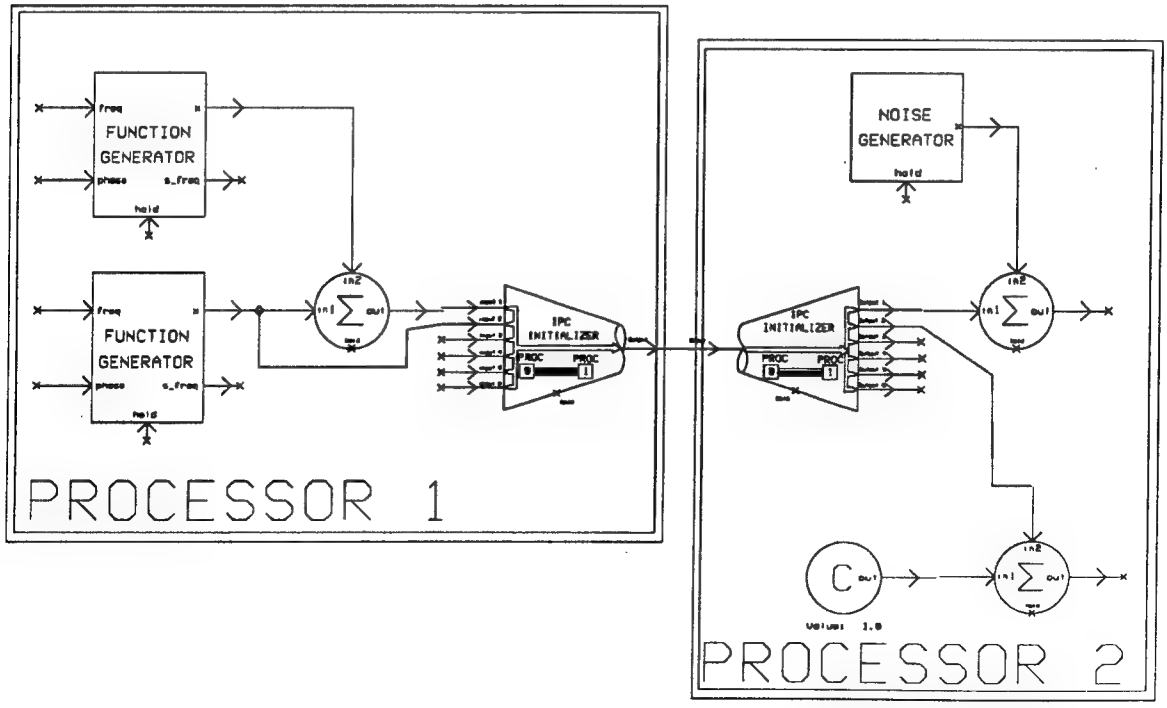

Figure 13: SYSTEM-2 with two data transfers used in Table 4

Chapter 6

Parallel Processing

Some of the features of the C40 such as dedicated parallel communications ports (commport) and DMA channels allow us to develop an efficient protocol. Each commport is bidirectional and can be used to send and receive data simultaneously. Port arbitration is done entirely by the C40 hardware and thus it is much simpler to transfer data in both directions. The total commport transfer rate of 20 MBytes per second allows for a high degree of parallel processing.

A test system for the CELP algorithm was implemented using three C40 processors as shown in Figure 6-1. Processor 1 handles the analog to digital and digital to analog conversions. A buffer of of frame of speech samples (size of 240) is created and transferred to the second processor which then encodes the samples into 144 bits. The encoded bits are then transferred to a third processor that decodes the bits and automatically transfers them (without any CPU intervention) to processor 1 which then performs the D/A conversion. A large delay (sum of the encoding and decoding delays) would be introduced if the D/A conversion required processor 1 to wait for the decoded frame. Instead, a constant delay of 1 frame is introduced between the input and the output of the first processor. The first output frame of speech then consists of all zeros after which the actual decoded speech signal is output.

All the data transfers are performed by the DMA co-processors. The inter-processor communication(IPC) algorithm is similar to that outlined in [13] and shown in Figure 6-2. However since the test system can withstand delays, the use of buffers provides a more efficient manner of performing the IPC. In particular, as shown in Figure 6-3, the use of two buffers that the CPU alternately writes to and the DMA reads from, increases the IPC efficiency tremendously since the CPU no longer needs to copy the output from one buffer to another before starting the DMA to transfer the data. This reduces the CPU and DMA dependence to a large extent since the DMA transfers a buffer while the CPU writes to another buffer at all times.

Our IPC algorithm was observed to transfer a frame of 240 speech samples from processor 1 to processor 2 in 32 CPU cycles (40 nanoseconds per cycle). Thus while the CELP encoder requires nearly 500,000 cycles per frame, only a negligible 32 cycles are needed for the data transfer between the processors.

One more SPW implementation of the test system is shown in Figure 6-4. Here instead of using a circular buffer on processor 1, multirate blocks are used for forming a vector of 240 samples from the input speech from the A/D and for forming the output speech from the decoder vector output. This eliminates the use of the impulse trains for timing purposes and consequently the system is more elegant and needs less computations.

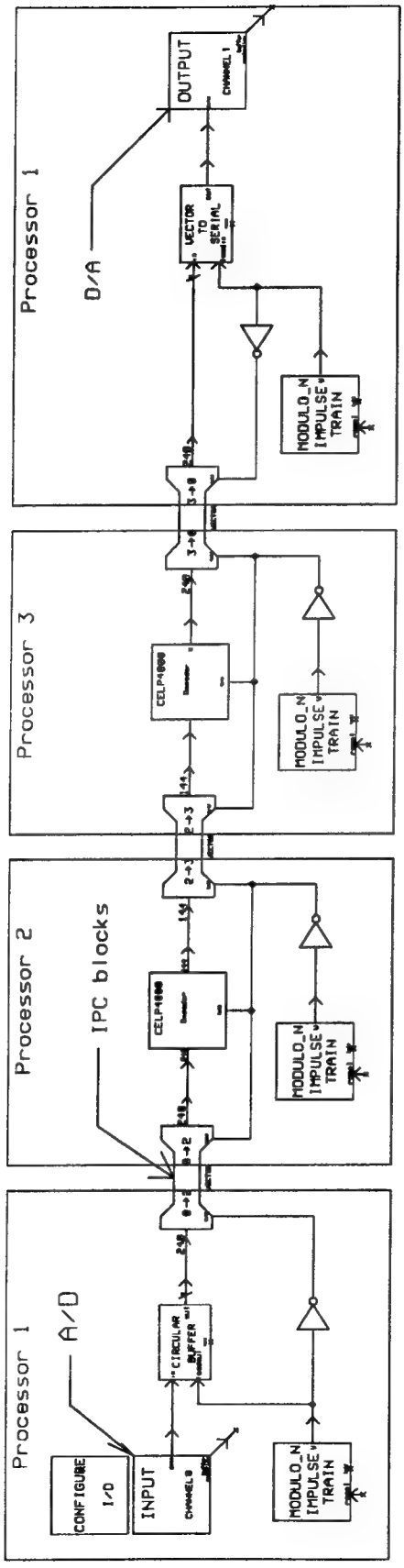

Figure 5.1: Block diagram of the overall CELP Test System

Figure 6-1:

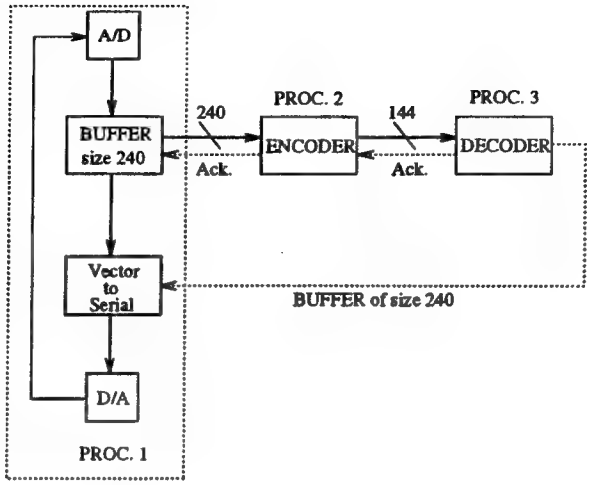

Figure 6-2: CPU/DMA actions per frame for CELP algorithm

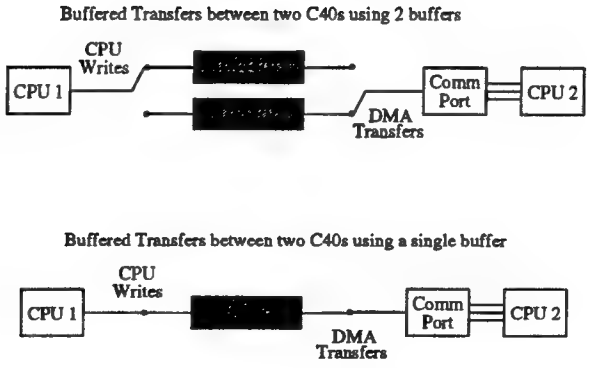

Figure 6-3: Use of buffers with the IPC algorithm

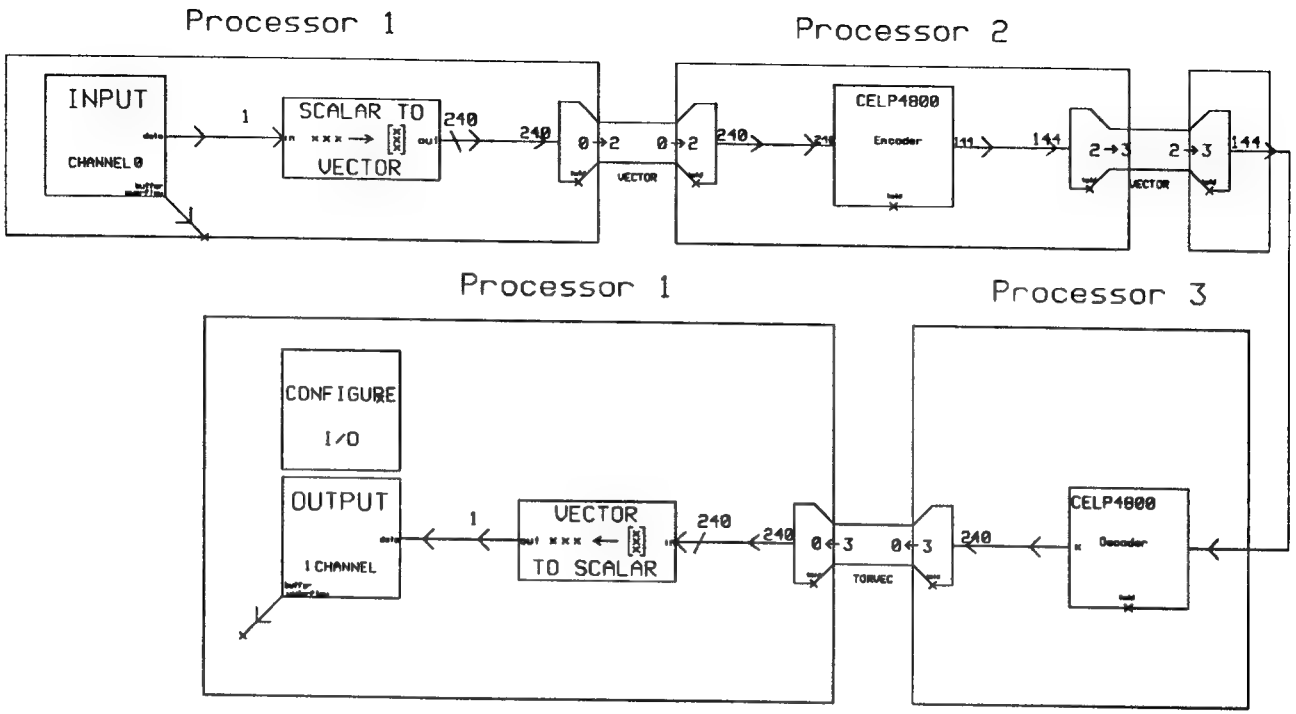

Figure 5.4: Block diagram of the CELP Test System with Multirate blocks

Chapter 7

Results

The CELP algorithm was implemented as rapidly prototypable blocks that can be called from the SPW library. Figure 6.1 and Figure 6.2 show the 4800 bps CELP encoder and decoder as they appear when called into a system. The encoder takes in speech in the form of a vector of length 240 (which corresponds to a 30 ms frame of speech sampled at 8 KHz) and outputs the encode vector of length 144 bits. The decoder on the other hand, takes in the encoded speech in the form of a length 144 vector and generates 240 samples of synthesized speech. The encoder and decoder were implemented as disparate blocks to preserve generality of use. The CELP coder operated successfully in real-time taking about 526,000 cycles for each frame. Since the time window available is 30 msec and with a processor cycle time of 40 nsec, the window yields a total of 750,000 available cycles. This means that the CELP coder operates about 70 % of the total time the system is run.

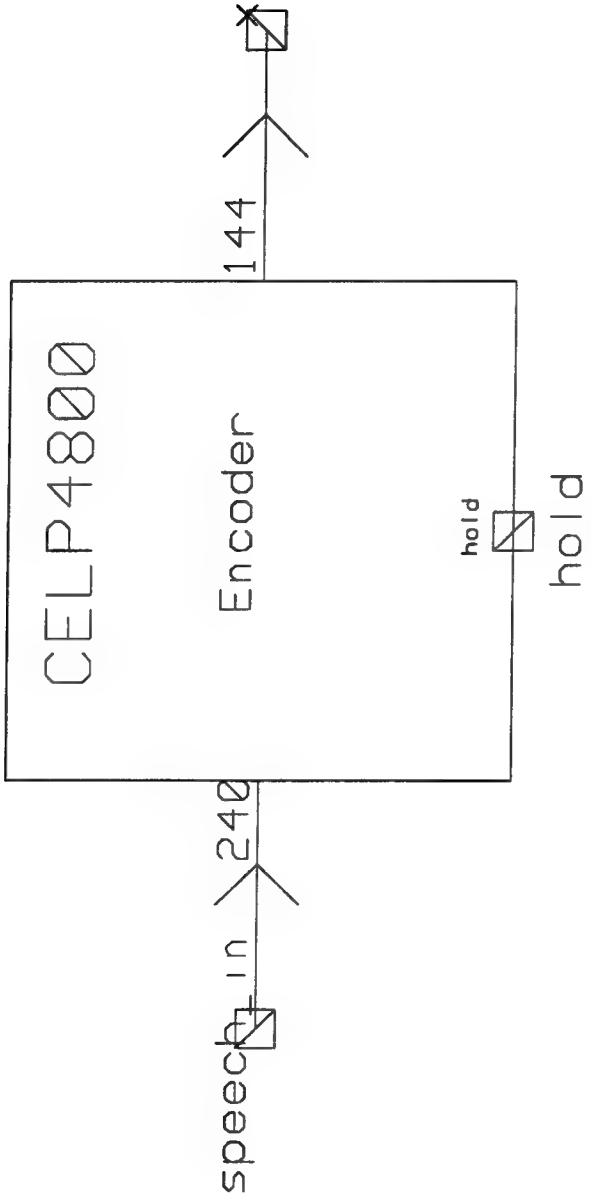

Figure 7-1: The CELP 4800 bps Encoder on SPW

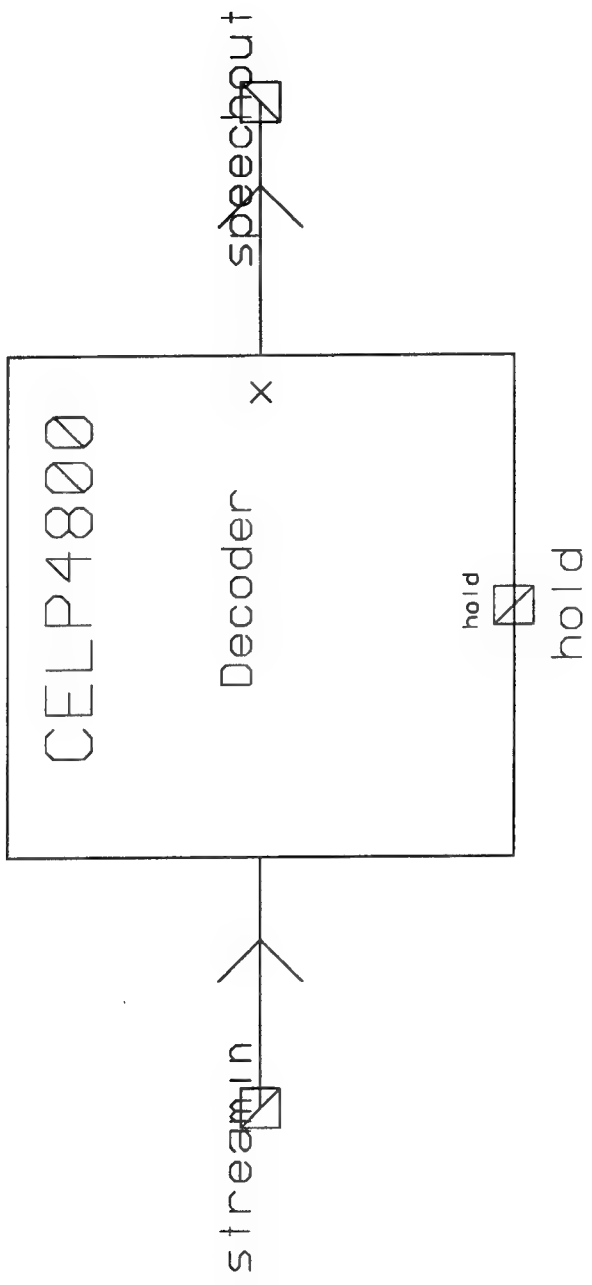

Figure 7-2: The CELP 4800 bps Decoder on SPW

Chapter 8

Conclusions

A real-time implementation of a 4800 bps CELP algorithm has been presented. The algorithm was implemented on a network of TMS320C40 processors. The encoder and decoder blocks were developed for use from the Signal Processing Worksystem (SPW) environment. A test system using multirate blocks along with the CELP encoder and decoder was implemented on three TMS320C40 processors. The quality of the synthesized speech was found to be greatly influenced by the noise on the A/D board. When a recorded speech file was sent through the system and the output played through the SUN audio port, the speech quality was found to be immensely better.

8.1 Current trends and outlook for the future

The present speech coders being used for digital cellular and PCS applications are derivatives of the 1016 Federal Standard 4800 bps CELP algorithm. To increase the quality of speech, the present systems have opted to go in for higher bit-rates. For example, the IS54 standard uses a form of CELP called the VSELP (Vector-Sum Excited Linear Prediction) at 7.9 kbps and the IS95 standards propose a similar speech coder at 7.2 kbps. GSM proposes a RPE-LTP (Residual Pulse

Excitation/ long term prediction) coder at 13 kbps. Qualcomm uses a variable rate vocoder (QCELP) which compresses 64 kbps speech to 8kbps through 1kbps. Further research is going on to reduce the bit-rate required to encode speech. Recently COMSAT has developed a 1200 bps speech coder. This algorithm is based on a tenth-order linear prediction analysis, split vector quantization of line spectral frequencies, differential pitch and gain quantization and adaptive postfiltering [18].

Bibliography

[1] Jayant, N. S. and Noll, P., Digital Coding of Waveforms, *Prentice-Hall, Englewood Cliffs, N.J.*, 1984.

[2] Atal, B. S., Predictive Coding of speech at low bit rates, *IEEE Trans. Commn. COM-30:600-614*, 1982.

[3] G. Davidson and A. Gersho, Complexity reduction methods for vector excitation coding, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1986, pp. 2379-2382.

[4] W. B. Kleign, D. J. Krasinski, and R. H. Ketchum, Improved speech quality and efficient vector quantization in CELP, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1988, pp. 155-158.

[5] L. A. Hernandez, F. J. Casajus-Quiros, A. R. Figueiras-Vidal and R. Garcia-Gomez, On the behaviour of reduced complexity code-excited linear prediction (CELP), *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1986, pp. 469-472.

[6] M. Copperi and D. Sereno, CELP coding for high-quality speech at 8 kbits/s, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1986, pp. 1685-1688.

[7] Hari Chakravarthula, S.Srinivas, G.Prescott, T.Johnson and Steve Taylor, Parallel Implementation of the CELP Speech-Compression Algorithm on a

network of TMS320C40 DSP processors, *International Conference on Signal Processing Applications and Technology (ICSPAT)*, Boston, USA, October '95.

[8] Joseph P. Campbell, Jr., Thomas E. Tremain and Vanoy C. Welch, The DoD 4.8 kbps Standard (Proposed Federal Standard 1016), *Advances in Speech Coding, Kluwer Academic Publishers*, 1991, Chapter 12, p.121-133.

[9] Joseph P. Campbell, Jr., Thomas E. Tremain and Vanoy C. Welch, An expandable error-protected 4800 bps CELP Coder, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1989, pp. 735-737.

[10] Campbell, J., V. Welch, and T. Tremain, The new 4800 bps Voice Coding Standard, *Proceedings of Military and Government Speech Tech*, 1989, pp. 64-70.

[11] B.S. Atal and M.R. Schroeder, Predictive Coding of Speech Signals and Subjective Error Criteria, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Vol. 66, pp. 1647-1652, Dec 1979.

[12] Chen, J. H. and A. Gersho, Real-Time Vector APC Speech Coding at 4800 bps with Adaptive Postfiltering, *Proc. Int. Conf. Acoust., Speech, Signal Processing*, 1987, pp. 2185-2188.

[13] G. Gupta, S. Srinivas, C. Neophytou, M. Krishnan, T. Johnson and S. Taylor, Design and Implementation of a Digital Spread Spectrum Wireless Modem using Rapid Prototyping Concepts, *Fifth International Conference on Signal Processing Applications & Technology (ICSPAT)*, Dallas,Texas, October '94.

[14] SPW - The DSP Framework, Signal Calculator Signal Flow Simulation, *COMDISCO Systems, Inc.*, March 1994.

[15] Texas Instruments Inc., TMS320 Floating-Point DSP Optimizing C Compiler User's Guide, 1993

[16] Texas Instruments Inc., TMS320 User's Guide, 1993.

[17] Chris Rowden, Speech Processing, *McGraw-Hill Book Company*, 1991.

[18] Ruth-Ann German, COMSAT's Home Page on the Web, *COMSAT Laboratories, URL- http://www.comsat.com*

*U.S. GOVERNMENT PRINTING OFFICE: 1996-509-127-47073

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.

	Unusual Time Intervals in Cycles
mpxfgen0.system 10000 iterations Threshold = 10000	279080, 13780, 251650, 51772, 178302
mpxfgen0.system 7000 iterations Threshold = 30000	267486, 53188, 45456, 2373836, 30876, 117934
mpxfgen0.system 25000 iterations Threshold = 60000	268300, 313648, 652516, 999202, 1111570, 340144, 359946, 184486, 193048, 766716
mpxfgen0.system 500000 iterations Threshold = 10^7	274906, 128640, 177950, 115960, 981606, 120918, 1219390, 281420, 130190

Table 2: Unusually high time intervals for mpxfgen0 system